

Wegweiser für Netzwerker

http://kickme.to/tiger/

Linux Wegweiser für Netzwerker

Wünschen Sie mehr Informationen zu der gedruckten Version des Buches "Linux: Wegweiser für Netzwerker" dann klicken Sie hier.

Inhaltsverzeichnis:

- Vorwort
- Einleitung
  - Informationsquellen
    - Linux-Dokumentation über FTP
    - Linux-Dokumentation über WWW
    - Kommerziell vorhandene Dokumentation
    - Usenet-Newsgruppen zu Linux
    - Linux Journal
    - Linux beziehen
  - Dateisystem-Standards
  - Über dieses Buch
  - Die offizielle gedruckte Version
  - Übersicht
  - Typografische Konventionen
  - Danksagungen
    - Die Ruhmeshalle
- Kapitel 1
Einführung in das Arbeiten mit Netzwerken
  - Historisches
  - UUCP-Netzwerke
    - Wie UUCP benutzt wird
  - TCP/IP-Netzwerke
    - Einführung in TCP/IP-Netzwerke
    - Ethernet

- Andere Arten von Hardware
- Das Internet-Protokoll (IP)
- IP über serielle Leitungen
- Transmission Control Protocol (TCP)
- User Datagram Protocol
- Mehr zu Ports
- Die Socket Library

Netzwerke unter Linux

- Verschiedene Streiflichter der Entwicklung
- Wo Sie den Kode herbekommen

Wie Sie Ihr System verwalten

- Systemsicherheit

Kapitel 2
Aspekte der Netzwerarbeit mit TCP/IP

- Netzwerk-Schnittstellen
- IP-Adressen
- Auflösung von IP-Adressen
- IP-Routing
  - IP-Netzwerke
  - Subnetze
  - Gateways
  - Die Routing-Tabelle
  - Metrische Werte
- Das Internet Control Message Protocol
- Das Domain Name System
  - Auflösung von Hostnamen
  - Einstieg in DNS
  - Namenssuche (Name Lookup) mit DNS
  - Domain-Name-Server
  - Die DNS-Datenbank
  - Reverse Lookup

Kapitel 3
Konfiguration der Hardware

- Geräte, Treiber und der ganze Rest
- Konfigurieren des Kernels
  - Kerneloptionen in Linux 1.0 und höher

Kerneloptionen in Linux 1.1.14 und höher

Eine Tour durch die Netzwerk-Geräte
Ethernet-Installation

Verkabelung
Unterstützte Karten
Ethernet-Autoprobing

Der PLIP-Treiber
Die SLIP- und PPP-Treiber

Kapitel 4
Konfiguration der seriellen Hardware

Software für Modem-Verbindungen
Einführung in serielle Geräte
Auf serielle Geräte zugreifen
Serielle Hardware

Kapitel 5
TCP/IP-Konfiguration

Einrichten des proc-Dateisystems
Installation der Programme
Noch ein Beispiel
Setzen des Hostnamens
IP-Adressen zuweisen
Einrichten von Subnetzen
Die Dateien hosts und networks
Schnittstellen-Konfiguration für IP

Die Loopback-Schnittstelle
Ethernet-Schnittstellen
Routing durch ein Gateway
Konfiguration eines Gateways
Die PLIP -- Schnittstelle
Die SLIP-Schnittstelle
Die Dummy-Schnittstelle

Alles über ifconfig
Testen mit netstat

Anzeigen der Routing-Tabellen
Anzeige der Interface-Statistiken
Aktive Verbindungen und Sockets

- Die ARP-Tabelle

Kapitel 6
Resolver und NameServer

- Die Resolver-Bibliothek
  - Die Datei host.conf
  - Konfiguration der Name-Server-Aufrufe -- resolv.conf
  - Robustheit des Resolvers
- Der Einsatz von named
  - Die Datei named.boot
  - Die Zonendateien
  - Die Master-Dateien
  - Test Ihrer DNS-Konfiguration
  - Andere Nützliche Dinge

Kapitel 7
Serial Line IP

- Allgemeine Anforderungen
- SLIP-Betrieb
- Handhabung privater IP-Netzwerke
- Verwendung von dip
  - Ein Beispiel-Script
  - dip-Referenz
    - Modembefehle
    - echo und term
    - Der get-Befehl
    - Der print-Befehl
    - Variablennamen
    - Die if- und goto-Befehle
    - send, wait und sleep
    - mode und default
- Der Server-Modus

Kapitel 8
Das Point-to-Point-Protokoll

- Die Ps entwirren
- PPP auf Linux
- Arbeiten mit pppd

- Arbeiten mit Optionsdateien
- Anwählen mit Chat
- Debuggen Ihres PPP-Setups
- IP-Konfigurations-Optionen
    - Wahl von IP-Adressen
    - Routen über einen PPP-Link
- Link-Kontrolloptionen
- Allgemeine Sicherheitsaspekte
- Authentizierung mit PPP
    - CHAP verglichen mit PAP
    - Die CHAP-Secrets-Datei
    - Die PAP-Secrets-Datei
- Konfiguration eines PPP-Servers

Kapitel 9
Wichtige NetzwerkFeatures

- Der inetd Super-Server
- Die tcpd-Zugriffs-Kontrolleinrichtung
- Die Dateien services und protocols
- Remote Procedure Call
- Konfiguration der r-Befehle

Kapitel 10
Das »Network Information System«

- Vertraut werden mit NIS
- NIS verglichen mit NIS+
- Die Client-Seite von NIS
- Betrieb eines NIS-Servers
- Sicherheit von NIS-Servern
- Mit NYS einen NIS-Client einrichten
- Die Wahl der richtigen Maps
- Verwendung von passwd- und group-Maps
- NIS und Shadow-Paßwörter verwenden
- Den traditionellen NIS-Kode verwenden

Kapitel 11
Das Network File System

- NFS vorbereiten
- Ein NFS-Volume mounten

- Die NFS-Dämonen
- Die exports-Datei
- Der Linux-Auto-Mounter

Kapitel 12
Verwalten von TaylorUUCP

- UUCP-Transfer und entfernte Programmausführung
  - Die interne Arbeitsweise von uucico
  - Kommandozeilen-Optionen von uucico
- UUCP-Konfigurations-Dateien
  - Eine sanfte Einführung in Taylor-UUCP
  - Was UUCP wissen muß
  - Site-Namen
  - Taylor-Konfigurations-Dateien
  - Allgemeine Konfigurations-Optionen -- die config-Datei
  - UUCP über andere Systeme informieren -- die sys-Datei
    - Systemname
    - Telefonnummer
    - port und speed
    - Der Login-Chat
    - Alternativen (Alternates)
    - Einschränken der Rufzeiten
  - Verfügbare Geräte -- die port-Datei
  - Wie eine Nummer gewählt wird -- die dial-Datei
  - UUCP über TCP
  - Verwendung einer direkten Verbindung
- Wer darf was bei UUCP -- Zugriffsrechte bestimmen
  - Befehlsausführung
  - Dateitransfer
  - Weiterleitung (Forwarding)
- Das Einwählen in Ihr System einrichten
  - getty einrichten
  - UUCP-Accounts einrichten
  - Wie Sie sich vor Schwindlern schützen
  - Seien Sie paranoid -- Rufsequenz-Prüfungen (Call Sequence Checks)
  - Anonymous UUCP
- Low-Level-Protokolle unter UUCP
  - Protokoll-Übersicht

        - Einstellen des Übertragungs-Protokolls
        - Bestimmte Protokolle wählen
    - Fehlersuche
    - Log-Dateien

Kapitel 13
Elektronische Post

    - Was ist denn nun eine Mail?
    - Wie wird Mail transportiert?
    - E-Mail-Adressen
    - Wie funktioniert Mail-Routing?
        - Mail-Routing im Internet
        - Mail-Routing in der UUCP-Welt
        - UUCP und RFC 822
    - Pfade und Landkarten
    - Die Konfiguration von elm
        - Globale elm-Optionen
        - Nationale Zeichensätze

Kapitel 14
smail zum Laufen bringen

    - UUCP-Setup
    - LAN-Setup
        - Schreiben der Konfigurations-Dateien
        - smail betreiben
    - Wenn Ihre Mail nicht durchkommt
        - smail kompilieren
    - Mail-Auslieferungsmodi
    - Verschiedene config-Optionen
    - Routing und Auslieferung von Nachrichten
    - Nachrichten routen
        - Die paths-Datenbank
    - Nachrichten an lokale Adressen ausliefern
        - Lokale Benutzer
        - Weiterleitung (Forwarding)
        - Alias-Dateien
        - Mailing-Listen

UUCP-basierte Transportarten
SMTP-basierte Transportarten
Qualifizierung von Hostnamen

Kapitel 15
Sendmail+IDA

Eine Einführung in Sendmail+IDA
Übersicht der Konfigurations-Dateien
Die Datei sendmail.cf
Eine sendmail.m4-Beispieldatei
Anwendungsspezifische sendmail.m4-Parameter
Punkte, die Pfade definieren
Definieren des lokalen Mailers
Mit gebouncter Mail umgehen
DNS-bezogene Punkte
Definition lokaler Systemnamen
UUCP-bezogene Punkte
Verteilerstationen und Mailer
Konfigurations-Tabellen
Die Hauptdatei Sendmail.mc
Welche Einträge werden denn nun wirklich benötigt?
Die Sendmail+IDA-Tabellen
mailertable
uucpxtable
pathtable
domaintable
aliases
Selten benutzte Tabellen
sendmail installieren
Auspacken der binären Distribution
Erzeugen von sendmail.cf
Testen der Datei sendmail.cf
Integrationstest zwischen sendmail.cf und den Tabellen
Administrivialitäten und dumme MailTricks
Weiterleitung von Nachrichten an einen Verteilerhost
Auslieferung von Mail an fehlerhaft konfigurierte Sites erzwingen
Auslieferung von Mail über UUCP erzwingen
Auslieferung von Nachrichten über UUCP verhindern

sendmail-Queue auf Anforderung ausführen
Ausgabe von Statistiken
Mischen und Abstimmen binärer Distributionen
Wo Sie weitere Informationen finden

Kapitel 16
Netnews
Geschichte des Usenet
Und was ist nun das Usenet?
Wie behandelt Usenet die News?

Kapitel 17
C News
News ausliefern
Installation
Die Datei sys
Die Datei active
Stapelverarbeitung von Artikeln (Batching)
News und Expiring
Weitere Dateien
Steuermeldungen
Die cancel-Meldung
newgroup und rmgroup
Die checkgroups-Meldung
sendsys, version und senduuname
C News in einer NFS-Umgebung
Verwaltungsaufgaben und -Tools

Kapitel 18
Eine Einführung in NNTP
Installation des NNTP-Servers
NNTP-Zugriff beschränken
NNTP-Autorisierung
Interaktion von nntpd und C News

Kapitel 19
Newsreader-Konfiguration
Konfiguration von tin

Konfiguration von trn
Konfiguration von nn

Anhang A
Ein Laplink-Kabel für PLIP

Anhang B
Beispiel-Konfigurationsdateien für smail

Anhang C
Copyright und Lizenzinformationen

Linux Network Administrator's Guide -- Copyright Information
GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Preamble
Terms and Conditions for Copying, Distribution, and Modification
Appendix: How to Apply These Terms to Your New Programs

The Berkeley Software Distribution Copyright

Anhang D
SAGE: Die Gilde der Systemadministratoren

Glossar

Bibliographie

Verwandte Literatur
HOWTOs
RFCs

Vorwort

Sie befinden sich gerade auf dem Weg in ein fantastisches Abenteuer in der Welt von Linux. Einfach ausgedrückt bringt Linux die Spannung und Leidenschaft zurück, die einmal vom Personal Computer ausging, als in seinen Kindheitstagen selbsternannte »Hacker«, mit Oszilloskop und Lötzinn bewaffnet, schlaflose Nächte damit zubrachten, ein lauffähiges System aus einem Bausatz zusammenzuflicken. Vielleicht haben sich die Werkzeuge und die Zeiten geändert (an die Stelle des Lötzinns ist der Debugger getreten), aber das Gefühl ist dasselbe. Es gibt immer noch Hacker, und Linux drückt deren Geist als komplettes UNIX-kompatibles Betriebssystem für den PC deutlich aus.

Trotz seiner Herkunft ist Linux nicht mehr nur etwas für abgedrehte Programmier-Freaks. Für Tausende von Leuten, die aus den verschiedensten Berufen stammen und mit unterschiedlichstem EDV-technischen Wissen ausgestattet sind, ist Linux zu dem Betriebssystem für Beruf, Erziehung und -- Okay, Okay -- einfach nur zum Spaß geworden.

Wie konnte Linux so mächtig und populär werden, wo es doch nicht von einem Team gutbezahlter (oder nicht so gutbezahlter) professioneller Programmierer entwickelt wurde? Haben wir hier vielleicht etwas, um das sich das FBI kümmern sollte?

Kooperation. Das allein ist der Grund dafür. Linux ist frei -- sowohl im Sinne von kostenlos als auch von »liberty« --, und diese Freiheit fördert positive Wechselwirkungen zwischen Programmierern, die den Traum von einem vollständig funktionierenden UNIX-Klon träumen. Fragen Sie einfach jeden Entwickler, warum er Stunde um Stunde mit dieser Zeitfalle namens Linux verbringt, und Sie werden immer dieselbe Antwort erhalten: ganz einfach aus Spaß am Hacken.

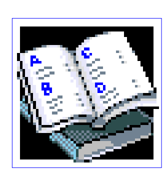
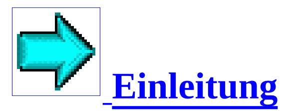
Was Sie aber vielleicht nicht wissen ist, daß, indem Sie Linux verwenden, Sie auch das Wachstum und die Entwicklung freier Software auf der ganzen Welt unterstützen. Freier Zugriff auf Software mit der Möglichkeit, diese zu modifizieren und ohne Einschränkungen weitergeben zu können, wird von vielen als fundamentales Recht aller Computerbenutzer betrachtet. (Zusammen mit dem Recht zu leben, dem Recht auf Freiheit und dem Streben nach mehr Plattenkapazität.) Software definiert, wie ein Computer verwendet wird, und Betriebssysteme sind dafür ein extremes Beispiel. Indem Sie ein bestimmtes Betriebssystem wählen, wählen Sie gleichzeitig ein Modell, nach dem die Maschine betrieben wird, angefangen vom Benutzer-Interface bis hin zum Gerätetreiber auf der untersten Ebene.

Der wichtigste Aspekt bei Linux ist, daß es von seinen und für seine Benutzer entwickelt und unterstützt wird. Nicht der Computermarkt, sondern die Benutzer bestimmen, in welche Richtung sich Linux entwickelt. Das ist ein weitaus greifbareres Ziel als das Streben nach Gewinn oder Marktanteilen. Obwohl das sicher bedeutet, daß Linux damit nicht jeden anspricht, kann das Publikum, für das es

gedacht ist, sein Verhalten doch direkt beeinflussen. Um es auf den Punkt zu bringen: Linux ist einfach unser Lieblings-Betriebssystem. Was kommt dagegen schon an?

Power to the people. Linux ist da.

Matt Welsh
Koordinator des Linux Documentation Project


Inhaltsverzeichnis   **Einleitung**

Einleitung

Nachdem das Internet in letzter Zeit ein richtiges Modewort geworden ist und auch sonst eher ernste Menschen eine Spritztour auf dem Information-Superhighway wagen, scheint die Vernetzung von Computern immer mehr den Status von Fernsehern und Mikrowellenherden anzunehmen. Dem Internet wird momentan eine ungewohnt hohe Aufmerksamkeit der Medien zuteil, und Soziologiestudenten fallen über Usenet-News-Gruppen her, um Forschungen über die neue »Internet-Kultur« anzustellen.

Natürlich gibt es Computernetze schon eine ganze Weile. Der Zusammenschluß mehrerer Computer zu einem lokalen Netzwerk ist selbst bei kleinen Installationen gängige Praxis, ebenso wie Verbindungen über längere Strecken mit Hilfe von Modems und Telefonleitungen. Das rapide wachsende Konglomerat weltweiter Netzwerke hat dazu geführt, daß der Einstieg in das globale Dorf auch für kleinere gemeinnützige Organisationen und Privatpersonen realisierbar ist. Die Einrichtung eines Internet-Host mit Mail und News, der Zugriff über Wählleitungen bietet, ist bezahlbar geworden, und mit der Einführung von ISDN (Integrated Services Digital Network) wird sich dieser Trend zweifelsohne noch beschleunigen.

Über Computernetzwerke zu sprechen, bedeutet häufig, über UNIX zu sprechen. Natürlich ist UNIX weder das einzige Betriebssystem mit Netzwerkfähigkeiten, noch wird es für ewig seine Vorreiterstellung behalten. Allerdings ist es schon seit einiger Zeit im Netzwerkgeschäft vertreten, und es sieht so aus, als würde das auch noch einige Zeit so bleiben.

Was UNIX für den privaten Benutzer so attraktiv macht ist die Tatsache, daß es zahlreiche Bemühungen gibt, freie UNIX-kompatible Betriebssysteme für PCs anzubieten. Beispiele hierfür sind 386BSD, FreeBSD -- und natürlich Linux.

Linux ist ein UNIX-Klon für Personal Computer, der frei verteilt werden kann. Momentan läuft es auf Intel 386-, 486- und Pentium-Maschinen, wird momentan aber auch auf andere Architekturen wie Motorola 680x0, DEC Alpha und MIPS portiert.

Linux wurde von einer großen Anzahl Freiwilliger über das Internet entwickelt. Das Projekt wurde 1990 von Linus Torvalds, einem finnischen Studenten, begonnen, der im Rahmen einer Vorlesung über Betriebssysteme damit anfing. Seit dieser Zeit ist Linux zu einem voll ausgestatteten UNIX-Klon herangewachsen, auf dem so verschiedene Anwendungen wie Emacs, das X-Window-System, TEX und eine ganze Reihe anderer Anwendungen laufen. Ein weites Spektrum an Hardware wird unterstützt, und Linux enthält eine vollständige Implementierung von TCP/IP, einschließlich SLIP und PPP. Linux ist mächtig, schnell und frei verfügbar, und seine Popularität nimmt auch außerhalb der Internet-Gemeinde rapide zu.

Das Linux-Betriebssystem selbst wird nach den Bestimmungen der »GNU General Public License« geschützt. Das ist dieselbe Copyright-Lizenz, wie sie für Software benutzt wird, die von der Free Software Foundation entwickelt wurde. Diese Lizenz erlaubt es jedem, die Software (kostenlos oder gegen eine Entlohnung) weiterzugeben und zu modifizieren, solange alle Modifikationen und Distributionen ebenfalls frei verteilt werden dürfen. Der Begriff »freie Software« steht hier für Freiheit, nicht nur für den Preis.

Linux kann zusammen mit anderen Betriebssystemen wie MS-DOS, OS/2 und so weiter auf Ihrer Maschine betrieben werden. Linux kann seine eigenen Partitionen verwenden und hat daher auf andere Betriebssysteme keinerlei Einfluß. Sie können einen Linux-Bootmanager installieren, mit dem Sie festlegen können, welches Betriebssystem gestartet wird, wenn das System gebootet wird.

Informationsquellen

Wenn Sie ein Neuling in der Linux-Welt sind, gibt es eine ganze Reihe von Ressourcen, die Sie sich ansehen sollten. Ein Internet-Anschluß ist zwar hilfreich, aber nicht unbedingt notwendig.

Handbücher des Linux-Dokumentationsprojekts

Das Linux-Dokumentationsprojekt (»Linux Documentation Project«, oder kurz LDP) ist eine Gruppe Freiwilliger, die eine Reihe von Handbüchern und Dokumenten zu Linux produzieren. Die Themen reichen dabei von der Installation bis hin zur Kernel-Programmierung. Folgende Handbücher sind vorhanden:

Linux Installation and Getting Started

von Matt Welsh. Das Buch beschreibt, wo Sie die Linux-Software bekommen und wie Sie sie installieren und benutzen können. Enthalten sind auch eine Einführung zu UNIX sowie Informationen zur Systemadministration, dem X-Window-System und zur Vernetzung.

Linux Kernel Hacker's Guide

von Michael K. Johnson. Dieses Buch behandelt die Interna des Linux-Systems. Es richtet sich an UNIX-Systemprogrammierer, die dem Linux-Kernel bestimmte Eigenschaften hinzufügen wollen.

Weitere Handbücher wie ein Benutzerhandbuch, ein Programmierhandbuch und ein Handbuch zur Systemadministration werden gerade entwickelt. Für weitere Informationen zum LDP können Sie sich über *mdw@sunsite.unc.edu* an Matt Welsh wenden.

INFO-SHEET und META-FAQ

Das sind zwei einführende Dokumente zum Linux-System. INFO-SHEET ist eine technische Übersicht des Linux-Systems, bei der die Eigenschaften und die vorhandene Software beschrieben werden. Die META-FAQ ist eine ausführliche Liste mit Verweisen auf weitere Informationen zu Linux -- Ihr Wegweiser durch das unüberschaubare Dickicht an Linux-Dokumentation.

HOWTO-Dokumente

Die Linux-HOWTOs sind eine Reihe von Dokumenten (alle ca. ein Dutzend Seiten stark), die sich mit verschiedenen Aspekten des Systems beschäftigen (beispielsweise mit der Installation und Konfiguration des X-Window-Systems oder der Verwendung des Kommunikationspakets term). Im allgemeinen sind sie im HOWTO-Unterverzeichnis der

nachfolgend aufgeführten FTP-Sites zu finden. Um sich einen Überblick der vorhandenen HOWTOs zu verschaffen, seien Sie auf die Bibliographie am Ende des Buches oder auf die Datei HOWTO-INDEX verwiesen.

Sie können sich das *Installation HOWTO* besorgen, das beschreibt, wie Sie Linux auf Ihrem System installieren können. Das *Hardware Compatibility HOWTO* enthält eine Liste aller Hardware-Komponenten, mit denen Linux arbeitet, und das *Distribution HOWTO* enthält eine Liste mit Software-Anbietern, die Linux auf Diskette oder CD-ROM verkaufen.

Linux Frequently Asked Questions

Die Linux Frequently Asked Questions (FAQ) enthält eine große Sammlung von Fragen zum System sowie die dazugehörigen Antworten. Dieses Dokument ist Pflichtlektüre für alle Einsteiger.

Linux Software Map

Eine Liste mit Software und Anwendungen, die unter Linux verfügbar sind. Obwohl diese Liste weit davon entfernt ist, vollständig zu sein (Tausende von Programmen können auf Linux portiert werden oder wurden es bereits), sind doch viele der größeren Anwendungen aufgeführt.

Linux-Dokumentation über FTP

Wenn Sie Zugriff über »anonymous FTP« haben, können Sie die gesamte Linux-Dokumentation über verschiedene Sites herunterladen, darunter

`sunsite.unc.edu:/pub/Linux/docs`

und

`tsx-11.mit.edu:/pub/linux/docs`

Diese Sites werden von einer Reihe von Sites auf der ganzen Welt gespiegelt.

Linux-Dokumentation über WWW

Viele Linux-basierte WWW-Sites sind vorhanden. Die Heimat-Site für das Linux Documentation Project kann über den folgenden URL erreicht werden:

`http://sunsite.unc.edu/mdw/linux.html`

Hypertext-Versionen vieler Linux-HOWTOs und anderer Dokumente sind ebenso vorhanden wie Verweise auf andere Linux-WWW-Archive.

Kommerziell vorhandene Dokumentation

Eine Reihe von Verlagen und Software-Anbietern vertreiben Linux-Software und die entsprechende Dokumentation im Versand. Anbieter in Deutschland sind u.a. LST (Lazarettstr. 8, 91054 Erlangen, 09131-8978-21), S.u.S.E. (Gebhardtstr. 2, 90762 Fürth, 0911-7405331, suse@suse.de) und Unifix

Software GmbH (Postfach 4918, 38039 Braunschweig, 0531-515161). Diese Unternehmen bieten *Linux Installation and Getting Started* und die Linux HOWTO-Dokumente in gedruckter und gebundener Form.

O'Reilly & Associates und O'Reilly/International Thomson Verlag veröffentlichen eine Serie mit Linux-Büchern, darunter auch den *Linux Network Administrator's Guide* (Deutsche Ausgabe: *Linux -- Wegweiser für Netzwerker*) aus dem Linux Documentation Project. Ebenfalls in dieser Reihe ist *Running Linux* (Deutsche Ausgabe: *Linux -- Wegweiser zur Installation und Konfiguration*) erschienen, ein Installations- und Benutzerhandbuch, in dem beschrieben wird, wie Sie mit Linux das meiste aus Ihrem Computer herausholen können.

Usenet-Newsgruppen zu Linux

Wenn Sie Zugriff auf Usenet-News besitzen, stehen Ihnen die folgenden Linux-Newsgruppen zur Verfügung:

comp.os.linux.announce

Eine moderierte Newsgruppe mit Ankündigungen neuer Software, Distributionen, Bugreports und Plänen der Linux-Gemeinde. Alle Linux-Benutzer sollten diese Gruppe lesen. Vorlagen können an *linux-announce@tc.cornell.edu* gemailt werden.

comp.os.linux.help

Allgemeine Fragen und Antworten zur Installation und zum Betrieb von Linux.

comp.os.linux.admin

Diskussionen zur Systemadministration unter Linux.

comp.os.linux.misc

Eine Newsgruppe mit verschiedenen Diskussionen zu Themen, die nicht in eine der obigen Kategorien fallen.

comp.os.linux.answers

In diese Gruppe werden regelmäßig die Linux-HOWTOs, diverse FAQs und andere regelmäßig erscheinende Informationen gepostet.

comp.os.linux.networking

Diskussionen zu Netzwerkfragen und -problemen unter Linux.

de.comp.os.linux.announce

Deutsche News-Gruppe für wichtige Ankündigungen zu Linux.

de.comp.os.linux

Deutsche News-Gruppe für Fragen und Diskussionen rund um Linux.

Neben den hier aufgezählten gibt es noch eine Reihe weiterer englischsprachiger News-Gruppen zum Thema Linux. Außerdem gibt es in einigen Ländern auch noch Linux-Gruppen in der jeweiligen Landessprache, z.B. *fr.comp.os.linux* in Frankreich.

Linux Journal

Das *Linux Journal* ist ein monatlich erscheinendes Magazin für die Linux-Gemeinde. Es wird von einer Anzahl von Linux-Aktivisten geschrieben und veröffentlicht. Die Themen der enthaltenen Artikel reichen von Fragen und Antworten für Einsteiger bis hin zu Interna der Kernel-Programmierung. Selbst wenn Sie Zugriff auf das Usenet haben, ist das Magazin eine gute Möglichkeit, mit der Linux-Gemeinde in Kontakt zu bleiben. Das *Linux Journal* wird von SSC, Inc. (P.O. Box 55549, Seattle, WA 98155, 206-FOR-UNIX, *sales@ssc.com*) herausgegeben.

Linux beziehen

Es gibt nicht eine einzelne Distribution der Linux-Software, sondern verschiedene, wie beispielsweise Slackware, Debian, Yggdrasil und viele mehr. Jede Distribution enthält alles, was Sie für den Betrieb eines kompletten Linux-Systems brauchen: den Kernel, grundlegende Dienstprogramme, Libraries, Support-Dateien und Anwendungsprogramme.

Linux-Distributionen können kostenlos über eine Reihe von Online-Quellen bezogen werden, darunter das Internet, verschiedene Bulletin-Board-Systeme oder Online-Dienste wie CompuServe und Prodigy. Im Internet können Sie Linux über verschiedene FTP-Archiv-Sites herunterladen. Die bekanntesten sind:

```
sunsite.unc.edu:/pub/Linux
tsx-11.mit.edu:/pub/linux
nic.funet.fi:/pub/OS/Linux
```

Die oben genannten Quellen für Linux-Informationen beschreiben, wie Sie Linux von anderen Orten beschaffen können.

Eine Reihe von Software-Anbietern vertreibt Linux auch auf Disketten, Tapes oder CD-ROMs. Das Linux-*Distribution HOWTO* enthält eine nahezu vollständige Liste. Die beiden bereits oben erwähnten Firmen SSC, Inc. und Linux Systems Labs vertreiben Linux ebenfalls auf CD-ROM.

Dateisystem-Standards

Ein großes Problem, das in der Vergangenheit Linux-Distributionen und separate Software-Pakete gleichermaßen betraf, war, daß es kein einheitlich akzeptiertes Layout des Dateisystems gab. Dies führte zu Inkompatibilitäten zwischen verschiedenen Paketen und konfrontierte Benutzer und Administratoren mit dem Problem, verschiedene Dateien und Programme finden zu müssen.

Um diese Situation zu verbessern, bildeten im August 1993 verschiedene Leute die »Linux File System Standard Group«, oder kurz FSSTND-Gruppe, die von Daniel Quinlan koordiniert wird. Nach sechsmonatiger Diskussion präsentierte die Gruppe einen Entwurf, der eine in sich schlüssige Struktur des Dateisystems darstellt und die Pfadnamen für die grundlegendsten Programme und Konfigurations-Dateien festlegt.

Dieser Standard ist bereits von den meisten der wichtigen Linux-Distributionen und -Pakete übernommen worden. In diesem Buch gehen wir daher davon aus, daß alle besprochenen Dateien sich an

den Stellen befinden, die vom Standard vorgegeben werden. Nur wenn eine lange Tradition existiert, die im Widerspruch zu dieser Spezifikation steht, werden Alternativen erwähnt.

Der Linux-Dateisystem-Standard kann über alle wichtigen Linux-FTP-Sites (und deren Mirrors) bezogen werden. Sie finden ihn etwa auf sunsite.unc.edu unter */pub/linux/docs*. Daniel Quinlan, der Koordinator der FSSTND-Gruppe, ist über *quinlan@yggdrasil.com* zu erreichen.

Über dieses Buch

Als ich im Jahr 1992 dem Linux Documentation Project beitrat, schrieb ich zwei kleine Kapitel über UUCP und *smail*, die ich dem System Administrator's Guide beisteuern wollte. Die Entwicklung von TCP/IP steckte noch in den Kinderschuhen, und als die »kleinen Kapitel« zu wachsen begannen, dachte ich laut darüber nach, daß es doch schön wäre, ein Netzwerk-Handbuch zu haben. »Großartig!« sagte jeder, »Setz dich dran!« Also setzte ich mich dran und schrieb die erste Version des »Networking Guide«, die ich im September 1993 veröffentlichte.

Das neue Netzwerk-Handbuch, das Sie gerade lesen, wurde von mir völlig neu geschrieben und schließt nun auch die neuen Anwendungen ein, die Linux-Benutzern seit der ersten Veröffentlichung zugänglich gemacht wurden.

Das Buch ist im großen und ganzen so organisiert, daß die von Ihnen durchzuführenden Schritte in der Netzwerk-Konfiguration Ihres Systems nacheinander durchlaufen werden. Es beginnt mit einer Diskussion grundlegender Netzwerk-Konzepte, wobei besonders auf TCP/IP-basierte Netzwerke eingegangen wird. Wir bahnen uns dann langsam unseren Weg von der TCP/IP-Konfiguration auf Geräteebene hin zum Setup gängiger Anwendungen wie *rlogin* und seiner Freunde, dem »Network File System« (NFS) und dem »Network Information System« (NIS). Dem folgt ein Kapitel darüber, wie Sie Ihre Maschine als UUCP-Knoten einrichten können. Der Rest des Buches widmet sich dann zwei Hauptanwendungen, die über TCP/IP und UUCP laufen: Elektronische Post und News.

Der E-Mail-Teil enthält eine Einführung in die intimeren Aspekte des Nachrichtentransports und -Routings sowie der zahllosen Adressierungs-Schemata, mit denen Sie konfrontiert werden könnten. Er beschreibt die Konfiguration und die Verwaltung von *smail*, einem Nachrichten-Transportsystem, das häufig bei kleineren Mail-Hubs verwendet wird. Besprochen wird auch *sendmail*, das für Leute gedacht ist, die ein komplizierteres Routing verwenden oder große Mengen von E-Mails verwalten müssen. Das Kapitel zu *sendmail* wurde von Vince Skahan geschrieben und beigesteuert.

Der News-Teil gibt einen Überblick darüber, wie Usenet-News arbeitet. Er behandelt C-News, die im Moment am weitesten verbreitete News-Transportsoftware. Besprochen wird auch die Verwendung von NNTP, über das News in lokalen Netzwerken gelesen werden können. Das Buch endet mit einem Kapitel über die Pflege und Fütterung der bekanntesten Linux-Newsreader.

Natürlich kann ein Buch niemals alle Fragen beantworten, die Sie möglicherweise haben. Seien Sie also geduldig, wenn Sie den Anweisungen dieses Buches folgen und dennoch nicht alles so funktioniert, wie Sie es erwarten. Einige Ihrer Probleme könnten durch dumme Fehler meinerseits entstanden sein, der Grund kann aber auch darin liegen, daß sich die Netzwerk-Software geändert hat. Daher sollten Sie zuerst einen Blick auf die aufgeführten Informationsquellen werfen. Die Chancen sind groß, daß Sie mit Ihrem Problem nicht alleine dastehen, so daß eine Korrektur oder zumindest eine Möglichkeit, den

Fehler zu umgehen, bekannt ist. Wenn Sie die Möglichkeit haben, sollten Sie sich auch immer die aktuellste Kernel- und Netzwerk-Release von einer der Linux-FTP-Sites oder einem BBS in Ihrer Nähe besorgen. Viele Probleme rühren von der Verwendung von Software aus verschiedenen Entwicklungsstadien her, die ein ordentliches Zusammenspiel verhindern. Schließlich befindet sich Linux ja immer noch »in Arbeit«.

Eine weitere gute Quelle, um sich über die aktuelle Entwicklung zu informieren, ist das Networking-HOWTO. Es wird von Terry Dawson(1) verwaltet und enthält die neuesten Informationen. Die jeweils aktuelle Version kann über die oben erwähnten FTP-Sites bezogen werden. Bei Problemen, die Sie auf andere Art nicht zu lösen vermögen, können Sie auch den Autor dieses Buches ansprechen. Bitte verzichten Sie aber darauf, die jeweiligen Entwickler direkt um Hilfe zu bitten. Diese widmen Linux bereits einen großen Teil ihrer Zeit und gelegentlich führen sie sogar ein Leben jenseits des Netzes. :-)

Die offizielle gedruckte Version

Im Herbst 1993 fragte mich Andy Oram, der sich so ziemlich von Anfang an in der LDP-Mailingliste tummelte, ob ich mein Buch nicht bei O'Reilly & Associates veröffentlichen lassen wollte. Darüber war ich sehr erfreut, denn ich hatte niemals erwartet, daß mein Buch so erfolgreich wird. Wir kamen schließlich überein, daß O'Reilly eine erweiterte offizielle gedruckte Version des Netzwerk-Handbuchs mit mir herausbringt, während ich das eigentliche Copyright behalte, so daß die Quellen dieses Buches frei verteilt werden können. Das bedeutet, daß Sie frei wählen können: Sie können die über das Netz vertriebenen LATEX-Quellen (bzw. die vorformatierten DVI- oder PostScript-Versionen) herunterladen und ausdrucken, oder Sie können die offizielle gedruckte Version von O'Reilly kaufen.

Warum sollten Sie also Geld für etwas ausgeben, was Sie anderswo umsonst bekommen können? Ist Tim O'Reilly von Sinnen, etwas zu veröffentlichen, was jeder andere ausdrucken oder sogar selbst weiterverkaufen darf?(2) Oder gibt es einen Unterschied zwischen diesen Versionen?

Die Antworten lauten »Es kommt darauf an«, »Nein, sicher nicht« und »Ja und Nein«. O'Reilly & Associates nimmt das Risiko der Veröffentlichung dieses Buches auf sich, aber ich hoffe, daß es sich für den Verlag auszahlt. Falls es das tut, glaube ich, daß dieses Projekt als Beispiel dafür dienen kann, wie die Welt der freien Software mit Unternehmen kooperieren kann, um etwas zu produzieren, was beiden Seiten Vorteile bringt. Aus meiner Sicht ist der große Dienst, den O'Reilly der Linux-Gemeinde erweist (neben der Tatsache, daß dieses Buch nun in Ihrem lokalen Buchladen zu finden ist), der, daß Linux dabei unterstützt wird, als etwas Ernstzunehmendes erkannt zu werden: eine mögliche und nützliche Alternative zu kommerziellen UNIX-Betriebssystemen für PCs.

Warum veröffentlichen sie es? Als einen Grund nannten sie mir, daß sie es als ihre Art von Buch betrachten. Es sei das, was sie erhoffen würden zu produzieren, würden sie mit einem normalen Autor darüber verhandeln, ein Buch über Linux zu schreiben. Der Inhalt, die Details und der Stil würden gut zu ihren anderen Angeboten passen.

Der Sinn der LDP-Lizenz ist sicherzustellen, daß niemand ausgeschlossen wird. Jeder kann sich eine Kopie dieses Buches ausdrucken, und niemand wird sich daran stören, wenn Sie sich eine dieser Kopien besorgen. Wenn Sie bislang aber keine Chance hatten, sich die O'Reilly-Version anzusehen, sollten Sie sich einmal in einer Buchhandlung oder bei Ihren Freunden umsehen. Wir denken, daß das, was Sie

sehen werden, Ihnen gefallen wird und daß Sie sich ein persönliches Exemplar kaufen werden.

Wo sind nun die Unterschiede zwischen der gedruckten und der Online-Version? Andy Oram hat große Anstrengungen unternommen, mein erstes Gestotter in etwas zu verwandeln, was es wert ist, gedruckt zu werden. (Er hat sich auch alle anderen Bücher angesehen, die vom Linux Documentation Project herausgebracht wurden, und versorgte die Linux-Gemeinde wo er konnte mit seinem professionellen Rat.)

Seit Andy damit begann, das Netzwerk-Handbuch korrekturzulesen und die Kopien zu bearbeiten, die ich ihm schickte, verbesserte es sich zusehends gegenüber dem, was es vor einem halben Jahr noch war. Ohne seine Arbeit wäre dieses Buch kaum das geworden, was es nun ist. Dasselbe gilt auch für Stephen Spainhour, der das Buch in die Form gebracht hat, die Sie jetzt sehen. All diese Arbeiten sind wiederum in die Online-Version eingeflossen, so daß es inhaltlich keine Unterschiede gibt.

Dennoch *ist* die O'Reilly-Version anders. Die Leute bei O'Reilly haben viel Arbeit in das »Look and Feel« dieses Buches gesteckt, so daß ein wesentlich ansprechenderes Layout dabei herauskam, als dies mit Standard-LATEX jemals möglich wäre. Zusätzlich hat Chris Reilley alle Abbildungen aus der ursprünglichen Netzwerkversion überarbeitet und eine Reihe weiterer Abbildungen erstellt. Er hat auf großartige Weise das visualisiert, was ich mit meinen amateurhaften XFIG-Zeichnungen ursprünglich erreicht zu haben glaubte. Chris Tong und Susan Reisler haben auch einen wesentlich verbesserten Index erstellt.

Übersicht

Das Kapitel 1, *Einführung in das Arbeiten mit Netzwerken* diskutiert die Geschichte von Linux und enthält grundlegende Informationen zu UUCP, TCP/IP, verschiedenen Protokollen, Hardware und Sicherheit. Die nächsten Kapitel behandeln die TCP/IP-Konfiguration von Linux sowie die Ausführung einiger wichtiger Hauptanwendungen. IP wird im Kapitel 2, *Aspekte der Netzwerarbeit mit TCP/IP* noch genauer behandelt, bevor wir uns die Hände beim Editieren von Dateien und ähnlichem dreckig machen. Wenn Sie bereits wissen, wie IP-Routing funktioniert und wie die Adreßauflösung durchgeführt wird, können Sie dieses Kapitel überspringen.

Das Kapitel 3, *Konfiguration der Hardware* bespricht die grundlegendsten Konfigurations-Aspekte wie die Kompilierung des Kernels und das Einrichten Ihrer Ethernet-Karte. Die Konfiguration Ihrer seriellen Schnittstellen wird separat im Kapitel 4, *Konfiguration der seriellen Hardware* besprochen, weil sie nicht nur für TCP/IP-Netzwerke, sondern auch für UUCP-Systeme von Bedeutung ist.

Das Kapitel 5, *TCP/IP-Konfiguration* hilft Ihnen dabei, Ihre Maschine für TCP/IP einzurichten. Es enthält Installations-Hinweise für alleinstehende Hosts, bei denen nur das Loopback aktiviert ist, aber auch für an ein Ethernet angeschlossene Hosts. Es werden auch einige nützliche Tools vorgestellt, mit denen Sie Ihr Setup testen und debuggen können. Das Kapitel 6, *Resolver und NameServer* diskutiert die Konfiguration der Hostnamen-Auflösung und beschreibt die Einrichtung eines Name-Servers.

Das Kapitel 7, *Serial Line IP* erklärt, wie eine SLIP-Verbindung hergestellt wird, und enthält einen ausführlichen Referenzteil zu *dip*, einem Tool, mit dem Sie den größten Teil der nötigen Schritte automatisieren können. Das Kapitel 8, *Das Point-to-Point-Protokoll* behandelt PPP und *pppd*, den

PPP-Daemon.

Das Kapitel 9, *Wichtige NetzwerkFeatures* enthält eine kurze Einführung zum Setup einiger der wichtigsten Netzwerk-Anwendungen wie *rlogin*, *rcp* etc. Dieses Kapitel beschreibt auch, wie Dienste vom *inetd*-Superserver verwaltet werden und wie Sie den Zugriff auf bestimmte sicherheitsrelevante Dienste auf eine bestimmte Gruppe von Hosts beschränken können.

Die Kapitel 10, *Das »Network Information System«* und Kapitel 11, *Das Network File System* diskutieren NIS und NFS. NIS ist ein nützliches Tool zur Verteilung administrativer Informationen, wie beispielsweise den Benutzer-Paßwörtern, in einem lokalen Netzwerk. NFS erlaubt die gemeinsame Nutzung von Dateisystemen zwischen verschiedenen Hosts in Ihrem Netzwerk.

Das Kapitel 12, *Verwalten von TaylorUUCP* gibt Ihnen eine ausführliche Einleitung in die Administration von Taylor-UUCP, einer freien Implementierung des UUCP-Pakets.

Der Rest des Buches befaßt sich ausführlich mit elektronischer Post und Usenet-News. Das Kapitel 13, *Elektronische Post* führt Sie in die zentralen Konzepte elektronischer Post ein. Dazu gehört beispielsweise, wie eine Postadresse aussieht und wie es ein Mail-System geregelt bekommt, Ihre Nachrichten an den Empfänger weiterzuleiten.

Die Kapitel 14, *smail zum Laufen bringen* und Kapitel 15, *Sendmail+IDA* decken das Setup von *smail* und *sendmail* ab, zwei Mail-Transportsysteme, die Sie unter Linux verwenden können. Das Buch behandelt beide, weil *smail* für den Anfänger leichter zu installieren ist, wohingegen *sendmail* flexibler ist.

Die Kapitel 16, *Netnews* und Kapitel 17, *C News* beschreiben, auf welche Weise News im Usenet verwaltet werden und wie Sie C-News installieren und verwenden können. C-News ist ein populäres Software-Paket zur Verwaltung von Usenet-News. Das Kapitel 18, *Eine Einführung in NNTP* beschreibt kurz, wie Sie einen NNTP-Daemon einrichten, damit in Ihrem lokalen Netzwerk auf News zugegriffen werden kann. Zum Schluß zeigt Ihnen das Kapitel 19, *Newsreader-Konfiguration*, wie Sie verschiedene Newsreader konfigurieren und verwalten können.

Typografische Konventionen

Im vorliegenden Buch werden folgende Auszeichnungen benutzt:

Fettdruck wird für die Namen von Rechnern, für die Benutzernamen und -IDs sowie gelegentlich zur Hervorhebung von Text benutzt.

Kursiv benutzen wir für Datei- und Verzeichnisnamen, für die Namen von Programmen und Befehlen, für Befehlszeilenoptionen, E-Mail-Adressen und Pfadnamen sowie zur Hervorhebung von neuen Begriffen.

`Sperrschrift`

wird in Beispielen benutzt, um den Inhalt von Dateien oder die Ausgaben von Befehlen darzustellen; außerdem für Environment-Variablen und Schlüsselwörter, die in Programmkode eingebettet sind.

`Sperrschrift kursiv`

benutzen wir für Optionen, Schlüsselwörter und solche Textstellen, die der Benutzer durch seine eigenen Texte ersetzen muß.

`Sperrschrift fett`

wird in Beispielen für Befehle und solche Textstellen gebraucht, die der Benutzer in genau dieser Form eingeben sollte.

Im gesamten Buch sind links neben dem Text verschiedene Icons zu finden.

Dieses Icon deutet auf Material hin, das im Gegensatz zu anderen UNIX-Systemen ganz spezifisch für die Installation und den Betrieb von Linux gilt. Der Vogel ist ein Sturmvogel, der eine Art Maskottchen des Linux-Projekts geworden ist.

Bei diesem Icon ist Vorsicht angesagt: Hier können Sie einen Fehler machen, der Ihrem System wehtun kann oder nur schwer zu beheben ist.

Danksagungen

Der in diesem Zusammenhang größte Dank geht an Vince Skahan, der das Kapitel 15, *Sendmail+IDA* schrieb. Vince hat seit 1987 eine große Anzahl von UNIX-Systemen administriert und betreibt momentan IDA *sendmail* auf ungefähr 300 UNIX-Workstations für über 2000 Benutzer. Er gibt zu, daß er ziemlich viel Schlaf versäumt hat, indem er einige *sendmail.cf*-Dateien »auf die harte Tour« editiert hat, bevor er 1990 IDA *sendmail* entdeckte. Er gibt auch zu, daß er schon sehnsüchtig auf die erste *perl*-basierte Version von *sendmail* wartet, um noch mehr obskuren Spaß zu haben…(3) Vince ist über *vince@halcyon.com* zu erreichen.

Dieses Buch verdankt viel auch den zahlreichen Menschen, die sich die Zeit genommen haben, es korrekturzulesen und viele Fehler auszubügeln, sowohl technische als auch grammatikalische. Der engagierteste von ihnen war Andy Oram von O'Reilly & Associates.

Ich schulde auch den Leuten bei O'Reilly vielen Dank, mit denen ich das Vergnügen hatte zu arbeiten: Stephen Spainhour, der das Buch überarbeitet und formatiert hat, um es in die nun vor Ihnen liegende Form zu bringen; Chris Reilley, der sich um die ganzen Abbildungen kümmerte; Edie Freeman und Jennifer Niederst, die das Cover, das Innenlayout und die Verwendung alter Holzschnitte als visuelles Thema (eine Idee von Lar Kaufman) entworfen haben; Susan Reisler und Chris Tong für die große Erweiterung des Index sowie dessen professionellere Gestaltung; Barbara Yoder für das mit mir arrangierte Interview für den O'Reilly-Katalog und zum Schluß natürlich Tim O'Reilly für die Courage, ein solches Projekt durchzuziehen.

Zutiefst verpflichtet bin ich auch Andres Sepúlveda, Wolfgang Michaelis, Michael K. Johnson und all den anderen Entwicklern, die ihre knappe Zeit geopfert haben, die im Netzwerk-Handbuch stehenden Informationen zu prüfen. Ich möchte auch all jenen danken, die die erste Version dieses Buches gelesen und mir Vorschläge und Korrekturen zugeschickt haben. Eine hoffentlich vollständige Liste aller ist in

der Datei *Thanks* der Online-Distribution zu finden. Letztendlich wäre dieses Buch nicht ohne die Unterstützung von Holger Grothe möglich gewesen, der mich mit der kritischen Internet-Connectivity versorgt hat.

Ich möchte auch den folgenden Gruppen und Unternehmen danken, die die erste Ausgabe meines Buches gedruckt und entweder mir oder dem Linux Documentation Project als Ganzem Geld gespendet haben: Linux Support Team, Erlangen, Germany; S. u.S.E. GmbH, Fuerth, Germany und Linux System Labs, Inc., Clinton Twp., USA.

Vince sagt: Dank an Neil Rickert und Paul Pomes für die viele Hilfe über die Jahre bei der Pflege und Fütterung von Sendmail+IDA sowie an Rich Braun für die ursprüngliche Portierung von Sendmail+IDA auf Linux. Der bei weitem größte Dank geht an meine Frau Susan für ihre Unterstützung bei diesem und anderen Projekten.

Die Ruhmeshalle

Neben den bereits genannten Leuten, die das Buch korrekturgelesen und mir Vorschläge und Korrekturen unterbreitet haben, gibt es eine große Anzahl von Leuten, die etwas zum Netzwerk-Handbuch beigesteuert haben. Ich bin ihnen sehr dankbar.

Nachfolgend eine Liste derer, deren Beiträge eine Spur in meinen Mail-Ordnern hinterließen. Ich entschuldige mich bei allen, denen ich an dieser Stelle nicht Dank sagen kann, weil ich einfach vergaß, ihre Mails aufzuheben.

Al Longyear, Alan Cox, Andres Sepulveda, Ben Cooper, Cameron Spitzer, D.J. Roberts, Emilio Lopes, Fred N. van Kempen, Gert Doering, Greg Hankins, Heiko Eissfeldt, J.P. Szikora, Johannes Stille, Karl Eichwalder, Les Johnson, Ludger Kunz, Marc van Diest, Michael K. Johnson, Michael Nebel, Michael Wing, Mitch D'Souza, Paul Gortmaker, Peter Brouwer, Peter Eriksson, Phil Hughes, Raul Deluth Miller, Rich Braun, Rick Sladkey, Ronald Aarts, Swen Thüemmler, Terry Dawson, Thomas Quinot, Yury Shevchuk.

Fußnoten

(1)

Terry Dawson ist unter terryd@extro.ucc.su.oz.au zu erreichen.

(2)

Denken Sie daran, daß Sie die Online-Version zwar ausdrucken dürfen, aber das O'Reilly-Buch nicht durch den Fotokopierer jagen, geschweige denn diese (hypothetischen) Kopien verkaufen dürfen.

(3)

Glaubst Du nicht, Vince, wir könnten es mit sed machen?

Inhaltsverzeichnis Vorwort Kapitel 1

Kapitel 1
Einführung in das Arbeiten mit Netzwerken

Historisches

Die Idee des Netzwerkes ist wahrscheinlich so alt wie die Telekommunikation selbst. Denken Sie an die Menschen, die in der Steinzeit gelebt haben, als Trommeln verwendet wurden, um Nachrichten auszutauschen. Stellen Sie sich vor, Höhlenmensch A möchte Höhlenmensch B einladen, sich gegenseitig mit Steinen zu bewerfen. Leider leben die beiden zu weit voneinander entfernt, als daß B hören könnte, was A trommelt. Welche Möglichkeiten hat A nun? Er kann 1. zu B hinübergehen, 2. sich eine größere Trommel besorgen oder 3. C bitten, der auf halber Strecke zwischen A und B wohnt, die Nachricht weiterzuleiten. Die letzte Möglichkeit wird Netzwerk genannt.

Natürlich haben die Geräte und Hilfsmittel unserer Vorfahren eine ziemliche Entwicklung durchgemacht. Heute unterhalten sich Computer über riesige Kabelstränge, Glasfasern, Mikrowellen und ähnliches, damit wir uns für das samstägliche Fußballspiel verabreden können.(1) In der folgenden Beschreibung wollen wir uns damit befassen, mit welchen Mitteln und auf welchen Wegen dies erreicht wird. Dabei lassen wir den Teil mit den Kabeln ebenso weg wie den mit dem Fußball.

In diesem Buch werden wir zwei verschiedene Arten von Netzwerken beschreiben: solche, die auf UUCP, und solche, die auf TCP/IP basieren. Dabei handelt es sich um Protokoll- und Softwarepakete, die die Mittel zum Datentransport zwischen zwei Computern zur Verfügung stellen. In diesem Kapitel sehen wir uns beide Netzwerkarten an und erläutern die zugrundeliegenden Prinzipien.

Wir definieren ein Netzwerk als eine Reihe von *Hosts*, die in der Lage sind, miteinander zu kommunizieren. Häufig sind diese Rechner von den Diensten einer Reihe von dedizierten Hosts abhängig, die Daten zwischen den Teilnehmern übertragen. Hosts sind häufig Computer, müssen es aber nicht unbedingt sein. Auch X-Terminals oder intelligente Drucker können Hosts sein. Ansammlungen von Hosts werden auch *Sites* genannt.

Kommunikation ist ohne so etwas wie eine Sprache oder einen Kode nicht möglich. In Computernetzwerken werden diese »Sprachen« kollektiv als *Protokolle* bezeichnet. Nun sollten Sie dabei nicht an geschriebene Protokolle denken, sondern vielmehr an eine sehr formalisierte Art des

Verhaltens, wie sie beispielsweise bei Staatsempfängen zu beobachten ist. In ähnlicher Weise sind auch die in Computernetzwerken verwendeten Protokolle nichts anderes als sehr strenge Regeln für den Austausch von Nachrichten zwischen zwei oder mehreren Computern.

UUCP-Netzwerke

UUCP ist die Abkürzung für »UNIX to UNIX Copy«. Es begann als Programmpaket zur Übertragung von Dateien über serielle Leitungen, zur Steuerung dieser Transfers und zur Ausführung von Programmen auf entfernten (remote) Sites. Seit seiner ersten Implementierung in den späten siebziger Jahren hat das System viele, auch schwerwiegende, Änderungen erfahren, dennoch sind die angebotenen Dienste eher spartanisch. Das Haupteinsatzgebiet liegt nach wie vor bei Wide-Area-Netzwerken, die auf Telefon-Wählleitungen basieren.

UUCP wurde 1977 zuerst von den Bell Laboratories für die Kommunikation zwischen deren UNIX-Entwicklungs-Sites entworfen. Mitte 1978 verband dieses Netzwerk bereits über 80 Sites. An Anwendungen bot es hauptsächlich E-Mail und Weiterleitungen von Druckaufträgen, allerdings lag die Hauptaufgabe des Systems im Verteilen neuer Software und Bugfixes. Heutzutage ist UUCP nicht mehr nur auf die UNIX-Umgebung beschränkt. Es existieren sowohl freie als auch kommerzielle Portierungen für eine ganze Reihe unterschiedlicher Plattformen wie AmigaOS, DOS, TOS, etc.

Einer der großen Nachteile von UUCP-Netzwerken ist die geringe Bandbreite. Auf der einen Seite schränken die Telefoneinrichtungen die maximale Übertragungsrate stark ein. Andererseits arbeiten UUCP-Links selten mit permanenten Verbindungen, statt dessen rufen sich die Hosts in regelmäßigen Intervallen über Wählleitungen an. Daher verbringt eine Nachricht, die über ein UUCP-Netz verschickt wird, die meiste Zeit damit, auf der Festplatte eines Host herumzuliegen und darauf zu warten, daß wieder eine Verbindung hergestellt wird.

Trotz dieser Einschränkungen gibt es immer noch viele UUCP-Netzwerke, die die ganze Welt umspannen. Betrieben werden sie meistens von Hobbyisten, die privaten Benutzern Netzwerkzugänge zu vernünftigen Preisen anbieten. Der Hauptgrund für die Beliebtheit von UUCP liegt sicher darin, daß es, im Vergleich zum Internet, so spottbillig ist. (Das ändert sich gerade ...) Um Ihren Computer in einen UUCP-Knoten zu verwandeln, benötigen Sie nur ein Modem, eine funktionierende UUCP-Implementierung und einen weiteren UUCP-Knoten, der bereit ist, Sie mit Mail und News zu versorgen.

Wie UUCP benutzt wird

Die UUCP zugrundeliegende Idee ist ziemlich einfach: Wie der Name es andeutet, werden grundsätzlich Dateien von einem Host auf den anderen kopiert. Sie können damit aber auch verschiedene Aktionen auf einem entfernten (»remote«) Host durchführen.

Stellen Sie sich vor, daß Ihre Maschine auf einen Host, nennen wir ihn einfach mal swim, zugreifen darf und dort den *lpr*-Befehl für Sie ausführen soll. Mit dem folgenden Befehl könnten Sie dieses Buch auf swim ausdrucken:(2)

```
$ uux -r swim!lpr !netguide.dvi
```

Mit *uux*, einem Befehl aus dem UUCP-Paket, wird ein *Job* auf swim initiiert. Der Job besteht aus der Eingabedatei *netguide.dvi* und der Anforderung, diese Datei an den *lpr*-Befehl zu übergeben. Mit der Option *-r* weisen Sie *uux* an, das entfernte System nicht direkt anzusprechen, sondern den Job zwischenzuspeichern, bis eine Verbindung hergestellt wurde. Diese Technik wird *Spooling* genannt.

UUCP erlaubt auch die Weiterleitung von Jobs und Dateien über mehrere Hosts hinweg, vorausgesetzt, daß diese kooperieren. Stellen Sie sich vor, daß swim über einen UUCP-Link zu groucho verfügt, der ein großes Archiv mit UNIX-Anwendungen zur Verfügung stellt. Um die Datei *tripwire-1.0.tar.gz* auf Ihre Site herunterzuladen, müssen Sie den folgenden Befehl eingeben:

```
$ uucp -mr swim!groucho!~/security/tripwire-1.0.tar.gz trip.tgz
```

Der erzeugte Job weist swim an, die Datei von groucho herunterzuladen und an Ihre Site zu übersenden, wo sie von UUCP in der Datei *trip.tgz* abgelegt wird. Gleichzeitig werden Sie über Mail benachrichtigt, daß die Datei eingetroffen ist. Das ganze läuft in drei Schritten ab. Zuerst wird der Job von Ihrem Rechner an swim geschickt. Sobald auf swim wieder eine Verbindung zu groucho besteht, wird die Datei heruntergeladen. Im letzten Schritt wird die Datei dann von swim auf Ihren Host übertragen.

Die heutzutage wichtigsten von UUCP-Netzwerken angebotenen Dienste sind Elektronische Post und News.

Elektronische Post -- kurz E-Mail -- erlaubt den Austausch von Nachrichten mit Benutzern auf anderen Hosts, ohne daß Sie wissen müssen, wie diese Hosts anzusprechen sind. Die Aufgabe der Weiterleitung einer Nachricht von Ihrem Host zum Zielsystem wird vollständig vom sog. Mail-Transport-System übernommen. In einer UUCP-Umgebung wird E-Mail üblicherweise mit Hilfe des *rmail*-Befehls übertragen. Das Programm wird dabei auf einem benachbarten Host ausgeführt, wobei ihm die Adresse des Empfängers und die eigentliche Nachricht übergeben werden. *rmail* übergibt die Nachricht dann an den nächsten Host und so weiter, bis das gewünschte Ziel erreicht ist. Wir werden dies im Kapitel 13, *Elektronische Post* detailliert behandeln.

News könnte man als eine Art verteiltes elektronisches Schwarzes Brett (Bulletin Board) beschreiben. Häufig bezieht sich der Ausdruck auf Usenet-News, dem bei weitem bekanntesten News-Netzwerk, mit einer geschätzten Teilnehmerzahl von 120.000 Sites. Der Ursprung des Usenet reicht bis ins Jahr 1979 zurück. Nachdem UUCP mit der neuen UNIX-V7-Version ausgeliefert worden war, hatten drei Studenten die Idee eines allgemeinen Informationsaustausches innerhalb der UNIX-Gemeinde. Sie stellten eine Reihe von Scripten zusammen, die zum ersten Netnews-System wurden. Im Jahr 1980 verband dieses Netzwerk die drei Sites: duke, unc und phs an zwei Universitäten in North Carolina. Daraus erwuchs schließlich das Usenet. Obwohl es als UUCP-basiertes Netzwerk begann, ist es nicht länger an einen bestimmten Netzwerktyp gebunden.

Die grundlegende Informationseinheit ist der Artikel. Sie können einen Artikel in eine Hierarchie von Newsgruppen einspeisen (»posten«), die bestimmten Themenbereichen gewidmet sind. Die meisten Sites empfangen nur einen Teil aller Newsgruppen, die durchschnittlich ca. 200 MB an Artikeln pro Tag umfassen.

In der UUCP-Welt werden News normalerweise über einen UUCP-Link verschickt, indem alle Artikel der angeforderten Gruppe zu einer Reihe von *Batches* zusammengepackt werden. Diese werden dann an

die empfangende Site geschickt, wo sie an den *rnews*-Befehl zum Entpacken und zur weiteren Bearbeitung übergeben

UUCP ist auch bei einer Reihe von Archiv-Servern das bevorzugte Medium. Sie erreichen diese Server üblicherweise, indem Sie sich über eine Telefonleitung mittels UUCP einwählen und sich als Gastbenutzer einloggen. Danach können Sie Dateien aus öffentlich zugänglichen Bereichen herunterladen. Solche Gastzugänge verwenden häufig uucp/nuucp als Loginnamen und Paßwort, oder etwas in der Art.

TCP/IP-Netzwerke

Obwohl UUCP eine durchaus vernünftige Wahl für das kostengünstige Anwählen von Netzwerkknoten darstellt, gibt es doch viele Situationen, in denen die verwendete Technik des Zwischenspeicherns und Weiterleitens nicht flexibel genug ist. Ein gutes Beispiel für die Grenzen von UUCP sind lokale Netzwerke (»Local Area Network«, oder kurz »LAN«). Solche lokalen Netzwerke bestehen aus einer kleinen Anzahl von Maschinen, die in einem Gebäude oder vielleicht auch nur auf einem Stockwerk stehen und untereinander verbunden sind, um eine homogene Arbeitsumgebung bereitzustellen. Typischerweise wollen Sie auf diesen Rechnern Dateien gemeinsam nutzen oder Anwendungen auf verschiedenen Maschinen ausführen.

Diese Aufgaben benötigen eine völlig andere Herangehensweise an Netzwerke. Statt komplette Dateien samt Job-Beschreibungen weiterzuleiten, werden alle Daten in kleinere Einheiten (Pakete) zerlegt, die dann unmittelbar an den Zielrechner weitergeleitet werden. Im Zielrechner werden die Pakete dann wieder zusammengesetzt. Ein solches Netzwerk wird als paketorientiertes Netzwerk (im Englischen *packet-switched*) bezeichnet. Unter anderem können Sie auf diese Weise interaktive Anwendungen über das Netzwerk ausführen. Der Preis für diese Möglichkeit ist die wesentlich erhöhte Komplexität der Software.

Die Lösung, die von UNIX-Systemen -- und vielen nicht-UNIX-Sites -- übernommen wurde, ist als TCP/IP bekannt. In diesem Abschnitt werden wir die darunterliegenden Konzepte betrachten.

Einführung in TCP/IP-Netzwerke

Die Wurzeln von TCP/IP liegen in einem Forschungsprojekt, das von der amerikanischen »Defense Advanced Research Projects Agency« (DARPA) im Jahre 1969 finanziert wurde. Was unter dem Namen ARPANET als experimentelles Netzwerk begann, wurde im Jahre 1975 in den normalen Betrieb übernommen, nachdem es sich als erfolgreich erwiesen hatte.

Im Jahre 1983 wurde das neue TCP/IP-Protokoll als Standard übernommen, und alle Hosts im Netzwerk mußten es von nun an einsetzen. Während ARPANET langsam zum Internet heranwuchs (wobei das ARPANET im Jahre 1990 zu existieren aufhörte), hatte sich TCP/IP schon auf Netzwerken außerhalb des Internet verbreitet. Herausragend sind dabei sicherlich lokale UNIX-Netzwerke, aber dank der zunehmenden Verbreitung schneller digitaler Telefoneinrichtungen wie ISDN hat TCP/IP auch eine vielversprechende Zukunft als Protokoll für Dialup-Netzwerke.

Als konkretes Beispiel für die Diskussion von TCP/IP in den folgenden Abschnitten führen wir die irgendwo in Fredland stehende Groucho-Marx-Universität (GMU) ein. Die meisten Institute betreiben

ein eigenes lokales Netzwerk, während andere eines gemeinsam nutzen, und wieder andere gleich mehrere betreiben. Alle sind aber über das Internet durch einen gemeinsamen Hochgeschwindigkeitslink miteinander verbunden.

Stellen Sie sich vor, Ihre Linux-Kiste ist mit einem LAN von UNIX-Hosts im mathematischen Institut verbunden (nennen wir sie einfach mal erdos). Um nun auf einen Host im physikalischen Institut, sagen wir auf quark, zugreifen zu können, müssen Sie den folgenden Befehl eingeben:

```
$ rlogin quark.physics
Welcome to the Physics Department at GMU
(ttyq2) login:
```

Wenn der Prompt erscheint, müssen Sie Ihren Loginnamen, z. B. andres, und Ihr Paßwort eingeben. Nach der korrekten Eingabe wird auf quark eine Shell gestartet, in der Sie arbeiten können, als würden Sie an der Konsole des Systems sitzen. Sobald Sie die Shell verlassen, erscheint wieder der Prompt Ihrer eigenen Maschine. Sie haben damit gerade eine der grundlegendsten auf TCP/IP zur Verfügung stehenden Anwendungen verwendet: remote Login, das Einloggen an einem entfernten Rechner.

Während Sie in quark eingeloggt sind, möchten Sie vielleicht eine X11-basierte Anwendung, beispielsweise ein Plot-Programm oder einen PostScript-Previewer, ausführen. Um nun der Anwendung mitzuteilen, daß die Fenster auf dem Bildschirm Ihres Rechners erscheinen sollen, müssen Sie die Umgebungsvariable DISPLAY setzen:

```
$ export DISPLAY=erdos.maths:0.0
```

Wenn Sie nun die Anwendung starten, wird Ihr eigener X-Server statt dem auf quark befindlichen angesprochen; alle Fenster erscheinen auf Ihrem Bildschirm. (Das setzt natürlich voraus, daß X11 auf erdos läuft.) Der Punkt ist hier, daß TCP/IP es quark und erdos erlaubt, X11-Pakete hin und her zu schicken, und Ihnen so vorgaukelt, daß Sie an einem System arbeiten. Das Netzwerk ist hier völlig transparent.

Eine weitere sehr wichtige Anwendung in TCP/IP-Netzwerken ist NFS, was für *Network File System* steht. NFS ist eine weitere Möglichkeit, Netzwerke transparent zu machen, weil es Ihnen erlaubt, Verzeichnishierarchien von anderen Hosts zu mounten. Diese erscheinen dann auf Ihrem Rechner so, als wären es lokale Dateisysteme. Beispielsweise könnten die Home-Verzeichnisse aller Benutzer auf einer zentralen Maschine gehalten werden, von dem aus alle anderen Hosts im LAN die Verzeichnisse mounten können. Der Effekt ist, daß sich Benutzer an jeder beliebigen Maschine einloggen können und sich dennoch immer in ihrem jeweiligen Home-Verzeichnis wiederfinden. Auf die gleiche Weise können Sie Anwendungen, die einen großen Platzbedarf haben (wie z. B. TEX), auf nur einem Rechner installieren und diese Verzeichnisse dann an andere Maschinen exportieren. Wir kommen noch in Kapitel 11, *Das Network File System* auf NFS zurück.

Das sind natürlich nur einige Beispiele für das, was Sie über TCP/IP-Netzwerke alles machen können. Die Möglichkeiten sind nahezu unbeschränkt.

Wir werfen nun einen genaueren Blick darauf, wie TCP/IP funktioniert. Diese Informationen helfen Ihnen zu verstehen, wie und warum Sie Ihre Maschine konfigurieren müssen. Wir beginnen mit einer Untersuchung der Hardware und arbeiten uns dann langsam vor.

Ethernet

Die in LANs am weitesten verbreitete Hardware ist im allgemeinen unter dem Namen *Ethernet* bekannt. Einfache Ethernets sind in der Installation relativ günstig, was, zusammen mit einer Übertragungsrate von 10 Megabit pro Sekunde, sicherlich für die große Popularität gesorgt hat.

Ethernet gibt es in drei Varianten: *Thick*, *Thin*, und *Zweidraht (»Twisted Pair«)*. Thin- und Thick-Ethernet verwenden beide ein Koaxialkabel, das sich aber jeweils in der Dicke und in der Art, wie ein Host an dieses Kabel angeschlossen wird, unterscheidet. Thin-Ethernet arbeitet mit einem T-förmigen BNC-Steckverbinder, den Sie in das Kabel einfügen und auf einen entsprechenden Stecker auf der Rückseite Ihres Computers aufstecken. Bei Thick-Ethernet müssen Sie ein kleines Loch in das Kabel drillen und einen Transceiver mit Hilfe eines sog. »Vampire Taps« einsetzen. Ein oder mehrere Hosts können dann mit diesem Transceiver verbunden werden. Bei Thin- und Thick-Ethernet kann das Kabel eine maximale Länge von 200 bzw. 500 Metern besitzen. Es wird daher auch 10base-2 oder 10base-5 genannt. »Twisted Pair« verwendet ein zweiadriges Kupferkabel und üblicherweise zusätzliche Hardware, die als *aktiver Hub* bekannt ist. Twisted Pair wird auch 10base-T genannt. Auch wenn das Hinzufügen eines weiteren Host bei einem Thick-Ethernet eine etwas haarige Angelegenheit ist, bringt es doch das Netzwerk nicht zum Absturz. Fügen Sie einen Host in ein Thinnet ein, legen Sie dagegen das Netzwerk zumindest für ein paar Minuten lahm, weil Sie das Kabel anschneiden müssen, um den Verbinder einzufügen.

Die meisten Leute ziehen Thin-Ethernet vor, weil es sehr billig ist. Steckkarten für den PC werden für unter DM 100 angeboten. Der Preis für Kabel liegt bei ein bis zwei Mark pro Meter. Bei größeren Installationen ist Thick-Ethernet die bessere Wahl. Beispielsweise wird am mathematischen Institut der GMU Thick-Ethernet verwendet, damit das Netzwerk nicht immer heruntergefahren werden muß, wenn ein neuer Rechner hinzugefügt wird.

Ein Nachteil der Ethernet-Technologie ist die begrenzte Länge der Kabel, was die Verwendung auf LANs beschränkt. Auf der anderen Seite können Sie aber mehrere Ethernet-Segmente miteinander verbinden, indem Sie sogenannte Repeater, Bridges oder Router verwenden. Repeater kopieren einfach die Signale zwischen zwei oder mehreren Segmenten, was bedeutet, daß alle Segmente zusammenarbeiten, als wäre es ein einzelnes Ethernet. Aufgrund der Timing-Anforderungen dürfen zwischen zwei Hosts im Netzwerk nicht mehr als vier Repeater verwendet werden. Bridges und Router sind da schon etwas aufwendiger. Sie analysieren die eingehenden Daten und leiten sie nur dann weiter, wenn sich der empfangende Host nicht im lokalen Ethernet befindet.

Ethernet arbeitet wie ein Bus-System, bei dem ein Host Pakete (oder *Frames*) mit einer Größe von bis zu 1500 Byte an einen anderen Host in demselben Ethernet übertragen kann. Ein Host wird dabei über eine sechs Byte lange Adresse angesprochen, die in die Firmware der Ethernet-Karte fest eingetragen ist. Diese Adressen werden üblicherweise als eine Sequenz von zweistelligen Hexadezimalzahlen geschrieben, die jeweils durch Doppelpunkte voneinander getrennt werden (beispielsweise aa:bb:cc:dd:ee:ff).

Ein von einem Rechner ausgesandter Frame wird von allen vorhandenen Rechnern registriert, aber nur der Ziel-Host liest das Paket und verarbeitet es. Versuchen zwei Stationen zur gleichen Zeit zu senden, tritt eine sog. *Kollision* auf. Diese Kollision wird dadurch aufgelöst, daß beide Stationen den Sendevorgang abbrechen und einige Augenblicke später wiederholen.

Andere Arten von Hardware

Bei größeren Installationen wie an der Groucho-Marx-Universität ist Ethernet üblicherweise nicht die einzig verwendete Ausrüstung. Bei der GMU sind die LANs aller Institute mit dem Universitäts-Backbone verbunden, einem Glasfaserkabel, über das FDDI (*Fiber Distributed Data Interface*) läuft. FDDI verwendet einen anderen Ansatz bei der Datenübertragung, bei dem eine Reihe sogenannter *Tokens* ausgesandt werden. Eine Station darf nur dann ein Paket übertragen, wenn sie eines dieser Tokens auffangen konnte. Die Hauptvorteile von FDDI liegen in der Geschwindigkeit von bis zu 100 Mbps und einer maximalen Kabellänge von bis zu 200 Kilometern.

Bei Langstrecken-Netzwerklinks wird häufig eine andere Technik verwendet, die auf einem Standard namens X.25 basiert. Viele der sogenannten öffentlichen Datennetze, wie Tymnet in den USA oder Datex-P in Deutschland, bieten diesen Service an. Für X.25 wird eine spezielle Hardware, namentlich ein »Packet Assembler/Disassembler« oder *PAD* benötigt. X.25 definiert selbst eine Reihe von Netzwerk-Protokollen, wird dessenungeachtet aber häufig genutzt, um TCP/IP-basierte Netzwerke mit auf anderen Protokollen basierenden Netzen zu verbinden. Weil IP-Pakete nicht einfach auf X.25-Pakete umgesetzt werden können (umgekehrt übrigens auch nicht), werden sie einfach in X.25-Pakete »eingepackt« und über das Netzwerk geschickt.

Öfter benutzen Amateurfunker ihre Geräte, um ihre Computer zu vernetzen. Diese Technik wird *Packet-Radio* oder *Ham-Radio* genannt. Das bei Ham-Radio verwendete Protokoll wird als AX.25 bezeichnet und ist von X.25 abgeleitet.

Andere Techniken verwenden langsame, aber billige serielle Leitungen für den Zugriff über Wählleitungen. Das erfordert wieder ein anderes Protokoll für die Übertragung von Paketen, zum Beispiel SLIP oder PPP, die wir später beschreiben werden.

Das Internet-Protokoll (IP)

Natürlich wollen Sie nicht, daß Ihr Netzwerk auf ein Ethernet beschränkt ist. Idealerweise sollten Sie ein Netzwerk ohne Rücksicht darauf verwenden können, welche Hardware es verwendet und aus wie vielen Untereinheiten es besteht. Beispielsweise finden Sie bei einer größeren Installation wie der Groucho-Marx-Universität üblicherweise eine ganze Reihe separater Ethernets, die auf irgendeine Art und Weise miteinander verbunden werden müssen. Am mathematischen Institut der GMU wird mit zwei Ethernets gearbeitet: ein Netzwerk mit schnellen Maschinen für Professoren und wissenschaftliche Mitarbeiter und eines mit langsamen Maschinen für die Studenten. Beide Netze hängen am universitätseigenen FDDI-Backbone.

Die Verbindung wird von einem speziellen Host, einem sogenannten *Gateway*, verwaltet, der ein- und ausgehende Pakete zwischen den beiden Ethernets und der Glasfaserleitung kopiert. Nehmen wir beispielsweise einmal an, Sie sitzen im mathematischen Institut und wollen von Ihrem Linux-Rechner aus auf quark im LAN des physikalischen Instituts zugreifen. Die Netzwerk-Software kann die Pakete nicht direkt an quark schicken, weil es sich nicht in demselben Ethernet befindet. Die Software muß sich nun also auf ein Gateway verlassen, das das Paket entsprechend weiterleitet. Das Gateway (nennen wir es mal sophus) leitet die Pakete an das Gateway niels weiter, das diese Funktion für das physikalische Institut übernimmt. Die Übertragung erfolgt über den Backbone, und niels liefert die Daten dann an den Zielrechner aus. Der Datenfluß zwischen erdos und quark ist in Abbildung 1--1 dargestellt.

Abbildung 1-1. *Die drei Schritte beim Senden eines Datagramms von erdos an quark*

Dieses Schema der Weiterleitung von Daten an einen entfernten Host wird *Routing* genannt, und die Pakete werden in diesem Zusammenhang häufig als *Datagramme* bezeichnet. Um die Dinge etwas zu vereinfachen, werden Datagramme über ein einziges Protokoll ausgetauscht, das von der verwendeten Hardware völlig unabhängig ist: IP, oder *Internet Protocol*. In Kapitel 2, *Aspekte der Netzwerarbeit mit TCP/IP* werden wir IP und Routing detaillierter behandeln.

Der Hauptvorteil von IP besteht darin, daß es physikalisch verschiedene Netzwerke zu einem scheinbar homogenen Netzwerk zusammenfaßt. Das wird als »Internetworking« bezeichnet, und das daraus resultierende »Meta-Netzwerk« wird *Internet* genannt. Beachten Sie an dieser Stelle den feinen Unterschied zwischen *einem* Internet und *dem* Internet. *Das Internet* ist der offizielle Name eines bestimmten globalen Internets.

Natürlich benötigt IP auch ein hardwareunabhängiges Adressierungsschema. Dies wird dadurch erzielt, daß jedem Host eine eindeutige 32-Bit-Zahl zugewiesen wird, die sog. *IP-Adresse*. Dargestellt wird eine IP-Adresse üblicherweise durch vier Dezimalzahlen, eine für jeden 8-Bit-Anteil, die durch Punkte voneinander getrennt sind. Zum Beispiel könnte quark die IP-Adresse 0x954C0C04 besitzen, die Sie dann als 149.76.12.4 schreiben würden. Dieses Format wird auch *Dotted Quad Notation* genannt.

Sie werden bemerkt haben, daß wir nun drei verschiedene Arten von Adressen haben: Zum einen haben wir Hostnamen wie z. B. quark, dann gibt es IP-Adressen, und zum Schluß gibt es noch Hardware-Adressen wie die 6-Byte-Ethernet-Adresse. Das alles muß irgendwie so zusammen passen, daß, wenn Sie rlogin quark eingeben, der Netzwerk-Software die IP-Adresse von quark übergeben werden kann. Liefert IP irgendwelche Daten an das physikalische Institut, muß es irgendwie herausfinden, welche Ethernet-Adresse zu welcher IP-Adresse gehört.

Mit dieser Thematik werden wir uns in Kapitel 2 ausführlicher befassen. Im Moment genügt es, sich zu merken, daß die für das Finden einer Adresse benötigten Schritte als *Auflösen des Hostnamens*, oder im Englischen als *Hostname Resolution*, bezeichnet werden, mit dem Ziel der Abbildung von Hostnamen auf IP-Adressen. *Address Resolution* ist dagegen der Prozeß, bei dem IP-Adressen auf Hardware-Adressen abgebildet werden.

IP über serielle Leitungen

Bei seriellen Leitungen gibt es einen als SLIP (*Serial Line IP*) bekannten »de facto«-Standard. Eine modifizierte Version von SLIP ist CSLIP, oder *Compressed SLIP*, bei der die IP-Header komprimiert werden, um die relativ geringe Bandbreite von seriellen Links besser auszunutzen. Ein anderes serielles Protokoll ist PPP, oder *Point-to-Point Protocol*. PPP bietet gegenüber SLIP eine ganze Reihe weiterer Features, einschließlich einer Überprüfung der Zugangsberechtigung. Der Hauptvorteil gegenüber SLIP liegt darin, daß es nicht auf den Transport von IP-Datagrammen beschränkt ist, sondern so entwickelt wurde, daß beliebige Datagramm-Arten übertragen werden können.

Transmission Control Protocol (TCP)

Mit dem Übertragen von Datagrammen von einem Host zum anderen ist es nicht getan. Wenn Sie sich in quark einloggen, wollen Sie eine zuverlässige Verbindung zwischen Ihrem *rlogin*-Prozeß auf erdos und dem Shell-Prozeß auf quark. Das bedeutet aber, daß die übertragenen Informationen vom Sender in Pakete aufgeteilt und vom Empfänger wieder zu einem richtigen Datenstrom zusammengesetzt werden müssen. So trivial Ihnen das auch vorkommen mag, so beinhaltet diese Aufgabe doch eine Reihe kritischer Aspekte.

Eine sehr wichtige Sache, die Sie über IP wissen sollten ist, daß es (absichtlich) nicht zuverlässig ist. Stellen Sie sich vor, daß 10 Leute in Ihrem Ethernet gleichzeitig anfangen, die neueste Release von XFree86 vom FTP-Server der GMU herunterzuladen. Die Menge der zu übertragenden Daten könnte für das Gateway zuviel sein, weil es zu langsam oder nicht ausreichend mit Speicher bestückt ist. Wollen Sie nun ein Paket an quark übertragen, könnten die Puffer von sophus gerade voll sein, und der Rechner wäre in diesem Fall nicht in der Lage, das Paket weiterzuleiten. IP löst dieses Problem, indem es dieses Paket einfach »vergißt«. Die Verantwortung für die Integritätsprüfung und die Vollständigkeit der Daten liegt daher bei den kommunizierenden Hosts, die entsprechend Sorge dafür tragen müssen, im Fehlerfall Datenpakete erneut zu senden.

Dies wird von wieder einem anderen Protokoll namens TCP, oder *Transmission Control Protocol*, erledigt, das einen zuverlässigen Dienst auf IP aufbaut. Das Hauptmerkmal von TCP ist, daß es IP verwendet, um bei Ihnen den Eindruck zu erwecken, daß eine einfache Verbindung zwischen den beiden Prozessen auf Ihrem Host und auf der entfernten Maschine existiert. Sie müssen sich also nicht darum kümmern, wie und auf welcher Route die Daten tatsächlich reisen. Eine TCP-Verbindung arbeitet grundsätzlich wie eine zwei-Wege-Pipe, bei der beide Prozesse schreiben und lesen können. Stellen Sie es sich wie ein Telefonat vor.

TCP identifiziert die Endpunkte einer solchen Verbindung anhand der IP-Adressen der beiden Hosts und der Nummer eines sog. *Ports* auf jedem Host. Ports können Sie sich als eine Art Zugangspunkt für Netzwerk-Verbindungen vorstellen. Wenn wir bei unserem Telefonbeispiel bleiben, können Sie IP-Adressen mit Vorwahlnummern (Nummern für bestimmte Städte) und Portnummern mit dem lokalen Kode (Nummern der individuellen Teilnehmer) vergleichen.

In unserem *rlogin*-Beispiel öffnet die Client-Anwendung (*rlogin*) einen Port auf erdos und stellt eine Verbindung mit Port 513 auf quark her, den der *rlogind*-Server verwendet. Auf diese Weise wird eine TCP-Verbindung hergestellt. Über diese Verbindung führt *rlogind* zuerst die Autorisierung durch und startet (»spawnt«) eine Shell. Die Standard-Ein- und Ausgabe der Shell wird in die TCP-Verbindung umgeleitet (»redirection«). Das bedeutet, daß alles, was Sie unter *rlogin* auf Ihrer Maschine eingeben, über den TCP-Stream geleitet und als Standardeingabe für die Shell verwendet wird.

User Datagram Protocol

Natürlich ist TCP nicht das einzige bei TCP/IP-Netzwerken verwendete Protokoll. Obwohl für Anwendungen wie *rlogin* geeignet, ist der zu verwaltende Overhead für manche Anwendungen völlig undenkbar. Statt dessen wird ein mit TCP verwandtes Protokoll namens UDP (*User Datagram Protocol*) verwendet. Genau wie TCP erlaubt auch UDP einer Anwendung, mit einem Dienst auf einem entfernten Rechner über einen bestimmten Port in Kontakt zu treten. Es wird aber keine Verbindung aufgebaut,

sondern es werden nur einzelne Pakete an den entsprechenden Dienst gesandt -- daher der Name.

Ein Dienst, der im allgemeinen UDP verwendet, ist NFS. (Manche NFS-Implementationen bieten alternativ auch eine TCP-Verbindung an; das wird aber nur selten genutzt.)

Mehr zu Ports

Sie können sich Ports als Zugangspunkte für Netzwerk-Verbindungen vorstellen. Will eine Anwendung einen Dienst anbieten, verwendet sie einen bestimmten Port und wartet auf Clients. (Dieses Warten ist auch als »Horchen« oder *Listening* am Port bekannt.) Möchte ein Client den Dienst nutzen, bestimmt er einen Port auf seinem lokalen Rechner und baut eine Verbindung zu dem Port des Servers auf dem entfernten Host auf.

Eine wichtige Eigenschaft von Ports ist, daß, wenn erst einmal eine Verbindung zwischen dem Client und dem Server aufgebaut wurde, sich jede Kopie des Servers an den Server-Port anhängen und nach weiteren Clients horchen kann. Das ermöglicht beispielsweise mehrere entfernte Logins, die alle Port 513 verwenden, gleichzeitig auf demselben Host. TCP ist in der Lage, die verschiedenen Verbindungen zu unterscheiden, weil sie alle von unterschiedlichen Ports oder Hosts kommen. Wenn Sie sich zum Beispiel zweimal von erdos aus in quark einloggen, verwendet der erste *rlogin*-Client den lokalen Port 1023 und der zweite Port 1022. Beide stellen aber eine Verbindung zu Port 513 auf quark her.

Dieses Beispiel zeigt die Verwendung von Ports als Treffpunkt, der von Clients angelaufen wird, wenn ein bestimmter Dienst in Anspruch genommen werden soll. Damit ein Client auch die richtige Portnummer anspricht, muß über diese Nummern eine Vereinbarung zwischen den Administratoren beider Systeme getroffen werden. Bei Diensten, die so weit verbreitet sind wie *rlogin*, werden die Nummern zentral verwaltet. Dies wird von der IETF (Internet Engineering Task Force) erledigt, die in regelmäßigen Abständen ein RFC namens *Assigned Numbers* veröffentlicht. Dieses Dokument beschreibt unter anderem die Portnummern der bekannten Dienste. Linux verwendet eine Datei namens */etc/services*, die Servicenamen mit Nummern verbindet.

Zu bemerken wäre noch, daß TCP und UDP zwar beide auf Ports aufbauen, diese Nummern aber nicht zu Konflikten führen. Das bedeutet, daß beispielsweise TCP-Port 513 mit UDP-Port 513 nicht identisch ist. Tatsächlich dienen diese Ports als Zugriffspunkte auf zwei verschiedene Dienste, nämlich *rlogin* (TCP) und *rwho* (UDP), um genau zu sein.

Die Socket Library

Beim UNIX-Betriebssystem ist die Software, die all diese Aufgaben und Protokolle durchführt, üblicherweise direkt in den Kernel integriert, und so ist es auch bei Linux. Die gängigste Programmierschnittstelle in der UNIX-Welt ist die *Berkeley Socket Library*. Der Name stammt von einer bekannten Analogie, die Ports als Steckdosen (Sockets) und den Verbindungsaufbau zu einem Port als »Einstecken« beschreibt. Die Bibliothek stellt eine *bind*-Funktion bereit, mit der Sie einen entfernten Host, ein Transport-Protokoll und einen Dienst spezifizieren können, zu dem ein Programm eine Verbindung herstellen oder an dem es horchen kann. Zu diesem Zweck stehen die Funktionen *connect*, *listen* und *accept* bereit. Die Socket-Bibliothek ist noch etwas allgemeiner gehalten, denn sie bietet nicht nur eine Klasse von TCP/IP-basierten Sockets (die *AF_INET*-Sockets), sondern auch eine Klasse, die Verbindungen verwaltet, die auf dem lokalen Rechner stattfinden (die *AF_UNIX*-Klasse). Einige

Implementierungen können auch mit anderen Klassen wie dem XNS-Protokoll (*Xerox Networking System*) oder X.25 umgehen. Unter Linux ist die Socket Library Teil der Standard C Library (*libc*). Zur Zeit werden neben TCP/IP- und UNIX-Sockets auch noch die Protokolle IPX (Novell), AX.25 und NET/ROM (Ham-radio) sowie Appletalk unterstützt.

Netzwerke unter Linux

Als Ergebnis der Bemühungen von Programmierern aus aller Welt wäre Linux ohne das globale Netzwerk nicht denkbar gewesen. Es ist also nicht weiter überraschend, daß bereits in den frühen Entwicklungsphasen verschiedene Leute damit begannen, es um Netzwerk-Fähigkeiten zu erweitern. Eine UUCP-Implementierung lief nahezu von Anfang an unter Linux. Die Arbeit an TCP/IP-basierter Netzwerktechnik begann ungefähr im Herbst 1992, als Ross Biro und andere das entwickelten, was später unter dem Namen Net-1 bekannt wurde.

Nachdem Ross im Mai 1993 mit der aktiven Entwicklung aufhörte, begann Fred van Kempen mit der Arbeit an einer neuen Implementierung, wobei größere Teile des Kodes neu geschrieben wurden. Diese Implementierung war unter dem Namen Net-2 bekannt. Die erste öffentliche Release, Net-2d, war im Sommer 1993 (als Teil des 0.99.10-Kernel) verfügbar und wurde seitdem von verschiedenen Leuten (besonders von Alan Cox) verwaltet und erweitert und ist nun als Net-2Debugged bekannt. Nach Behebung vieler Fehler und nach zahlreichen Verbesserungen des Kodes wurde es in Net-3 umbenannt, nachdem Linux 1.0 verfügbar war. Diese Version des Netzwerk-Kodes ist im Moment auch in den offiziellen Kernel-Releases enthalten.

Net-3 bietet Gerätetreiber (»Device Driver«) für eine große Anzahl unterschiedlicher Ethernet-Karten, aber auch SLIP (zur Übertragung von Daten über serielle Leitungen) und PLIP (für parallele Leitungen). Mit Net-3 verfügt Linux über eine TCP/IP-Implementierung, die sich in lokalen Netzwerken sehr gut verhält. Der gezeigte Durchsatz schlägt sogar einige der kommerziellen UNIX-Versionen für PCs. Nachdem in der Kernelreihe 1.1.* die Stabilisierung des Kode im Vordergrund stand, geht die Entwicklung seit 1.3.1 wieder verstärkt in Richtung einer Erweiterung der vorhandenen.

Darüber hinaus gibt es verschiedene Projekte, die die Vielseitigkeit von Linux erhöhen sollen. Ein PPP-Treiber (das Point-to-Point-Protokoll, ein anderer Weg, Netzwerkpakete über serielle Leitungen zu übertragen) ist im Moment im Beta-Stadium. Treiber für Ham-Radio-Protokolle AX.25 und Netrom sowie Appletalk befinden sich im Alpha-Stadium. Alan Cox hat auch einen Treiber für das IPX-Protokoll von Novell implementiert, diese Bemühungen liegen augenblicklich aber leider auf Eis, weil Novell nicht bereit ist, die notwendige Dokumentation für die darüberliegenden Protokollschichten zur Verfügung zu stellen. Eine weitere, sehr vielversprechende Unternehmung ist *samba*, ein freier NetBIOS-Server für UNIX-Systeme, geschrieben von Andrew Tridgell.(3)

Verschiedene Streiflichter der Entwicklung

Fred setzte die Entwicklung mit Net-2e fort, das ein sehr überarbeitetes Design des Netzwerk-Layers aufwies. Allerdings hat man später nicht mehr viel von Net-2e gehört.

In absehbarer Zukunft wird uns aber wohl Net-3 erhalten bleiben. Alan arbeitet momentan auf Hochtouren an Erweiterungen wie IP Multicasting und verbesserter Unterstützung für UNIX-Sockets

und das IPv6-Protokoll, dem zukünftigen Nachfolger von IP.

Auch wenn die verschiedenen Netzwerk-Implementierungen alle bestrebt sind, dieselben Dienste anzubieten, so gibt es auf Kernel- und Treiberebene doch erhebliche Unterschiede. Aus diesem Grund können Sie auch ein System, das mit einem Net-2e-Kernel arbeitet, nicht mit den Utilities von Net-2d oder Net-3 konfigurieren und umgekehrt. Diese Aussage gilt nur für Befehle, die intern sehr eng mit dem Kernel zusammenarbeiten. Anwendungen und gängige Netzwerkbefehle wie *rlogin* oder *telnet* laufen mit jedem Kernel.

Dennoch sollten Sie sich von den verschiedenen Netzwerkversionen nicht verwirren lassen. Solange Sie nicht an der aktiven Entwicklung teilnehmen, müssen Sie sich nicht darum kümmern, welche Version des TCP/IP-Kodes Sie benutzen. Parallel zu den offiziellen Kernel-Releases gibt es immer auch Netzwerk-Utilities, die mit dem im Kernel integrierten Netzwerk-Kode kompatibel sind.

Wo Sie den Kode herbekommen

Die neueste Version des Linux-Netzwerk-Quellkodes können Sie über »anonymous FTP« von verschiedenen Servern herunterladen. Die offizielle FTP-Site für Net-3 ist sunacm.swan.ac.uk, gespiegelt von sunsite.unc.edu unter *system/Network/sunacm*, das wiederum von einer Reihe deutscher Server wie ftp.gwdg.de oder ftp.uni-erlangen.de gespiegelt wird.

Die neuesten Kernel finden Sie auf nic.funet.fi in */pub/OS/Linux/PEOPLE/Linus*; sunsite und tsx-11.mit.edu spiegeln dieses Verzeichnis.

Wie Sie Ihr System verwalten

In diesem Buch befassen wir uns hauptsächlich mit Installations- und Konfigurationsfragen. Administration bedeutet aber wesentlich mehr als das -- nachdem Sie einen Dienst eingerichtet haben, müssen Sie ihn auch am Laufen halten. Den meisten Diensten müssen Sie nicht besonders viel Aufmerksamkeit schenken, während andere wie Mail und News die Erledigung von Routineaufgaben von Ihnen verlangen, um Ihr System auf dem neuesten Stand zu halten. Diese Aufgaben werden wir in späteren Kapiteln behandeln.

Das absolute Verwaltungsminimum ist die regelmäßige Prüfung der System- und der einzelnen Anwendungs-Logdateien auf Fehler und ungewöhnliche Ereignisse. In der Regel erledigen Sie dies durch das Schreiben einer Reihe administrativer Shell-Scripten, die periodisch von *cron*aus ausgeführt werden. Die Source-Distributionen einiger bedeutender Anwendungen wie *smail* oder C-News enthalten solche Scripten. Sie müssen diese entsprechend Ihren Anforderungen und Bedürfnissen anpassen.

Die Ausgabe von jedem Ihrer *cron*-Jobs sollte per E-Mail an einen administrativen Account gesandt werden. Per Voreinstellung senden viele Anwendungen Fehlerreports, Benutzungsstatistiken oder Zusammenfassungen von Logdateien an den root-Account. Das macht nur Sinn, wenn Sie sich gelegentlich als root einloggen; besser ist es dagegen, die Mail von rootan Ihren persönlichen Account weiterzuleiten, indem Sie ein Mail-Alias einrichten, wie dies in Kapitel 14, *smail zum Laufen bringen* beschrieben ist.

Gleichgültig, wie sorgfältig Sie Ihre Site konfiguriert haben, Murphys Gesetz garantiert, daß ein Problem

auftauchen *wird*. Darum bedeutet die Verwaltung eines Systems auch, daß Sie für Beschwerden verfügbar sein müssen. Normalerweise erwarten die Leute, daß der Systemadministrator zumindest als *root* per E-Mail erreichbar ist, es existieren aber auch andere Adressen, die häufig verwendet werden, um eine Person zu erreichen, die für einen bestimmten Aspekt der Verwaltung zuständig ist. Beispielsweise werden Beschwerden über eine fehlerhafte Mail-Konfiguration üblicherweise an *postmaster* adressiert. Probleme mit dem News-System werden dem Benutzer, *newsmaster* oder *usenet* mitgeteilt. Mail an *hostmaster* sollte an die Person weitergeleitet werden, die sich um die grundlegenden Netzwerkdienste und den DNS-Service (falls vorhanden) kümmert.

Systemsicherheit

Ein anderer wichtiger Aspekt der Systemadministration in einer Netzwerkumgebung ist der Schutz Ihres Systems vor Eindringlingen. Nachlässig verwaltete Systeme bieten böswilligen Menschen viele Ziele. Die Angriffe reichen von geknackten Paßwörtern bis hin zum Ethernet-Snooping, und die verursachten Schäden reichen von gefälschten Mails über Datenverlust bis hin zur Verletzung Ihrer Privatsphäre. Wir werden auf die Probleme im einzelnen dort eingehen, wo die Probleme auch wirklich auftreten. Gleichzeitig stellen wir Ihnen einige gängige Schutzmaßnahmen dagegen vor.

Dieser Abschnitt beschreibt einige Beispiele und grundlegende Techniken zur Systemsicherheit. Natürlich können die behandelten Aspekte nicht alle Sicherheitsthemen im Detail erläutern, denen Sie begegnen könnten; sie dienen bloß dazu, die möglichen Probleme zu verdeutlichen, die auftreten könnten. Aus diesem Grund ist das Lesen eines guten Buches über Sicherheit ein absolutes Muß, besonders bei einem vernetzten System.

Die Systemsicherheit beginnt mit guter Systemadministration. Das beinhaltet die Überprüfung von Eigentums- und Zugriffsrechten aller aktiven Dateien und Verzeichnisse, die Überwachung privilegierter Accounts etc. Beispielsweise überprüft das Programm COPS Ihr Dateisystem und gängige Systemdateien auf ungewöhnliche Rechte und andere Anomalien. Es ist auch klug, ein Paßwort-Paket zu verwenden, das verschiedene Regeln auf das Paßwort des Benutzers anwendet, um es so schwer wie möglich zu machen, es zu erraten. Das Shadow-Paßwort-Paket beispielsweise verlangt, daß ein Paßwort aus mindestens fünf Zeichen besteht und sowohl groß- als auch kleingeschriebene Buchstaben (und Zahlen) enthält.

Wenn Sie einen Dienst über das Netzwerk verfügbar machen, sollten Sie sicherstellen, daß Sie ihm die »geringsten Privilegien« zuweisen, d. h. ihm keine Dinge erlauben, die für seine Arbeit nicht benötigt werden. Zum Beispiel sollten Programme die UID nur, wenn unbedingt nötig, auf root oder einen anderen privilegierten Account einstellen dürfen. Zum Beispiel sollten Programme nur dann unter root-Berechtigung laufen wenn unbedingt nötig. Wenn ein Dienst nur für eine sehr eingeschränkte Anwendung eingesetzt werden soll, sollten Sie nicht zögern, ihn so restriktiv zu konfigurieren, wie es diese spezielle Anwendung zuläßt. Sollen andere beispielsweise direkt von Ihrer Maschine booten können, müssen Sie TFTP (Trivial File Transfer Protocol) installieren, damit sie grundlegende Konfigurationsdateien aus Ihrem */boot*-Verzeichnis herunterladen können. Wird es nun aber ohne Beschränkungen verwendet, erlaubt TFTP jedem Benutzer überall auf der Welt jede allgemein lesbare Datei von Ihrem System herunterzuladen. Sollte dies nun nicht in Ihrem Sinne sein, warum also nicht den TFTP-Dienst auf das */boot*-Verzeichnis beschränken?(4)

Diesen Gedanken weiterspinnend, könnten Sie verschiedene Dienste auch auf die Benutzer bestimmter

Hosts wie beispielsweise Ihres lokalen Netzwerks einschränken. In Kapitel 9, *Wichtige NetzwerkFeatures* stellen wir *tcpd* vor, das das für eine ganze Reihe von Netzwerk-Anwendungen tut.

Ein weiterer wichtiger Punkt ist die Vermeidung »gefährlicher« Software. Natürlich kann jede Software gefährlich sein, weil sie fehlerhaft sein kann, was clevere Leute ausnutzen könnten, um sich Zugang zu Ihrem System zu verschaffen. Solche Dinge kommen vor, und es kann keinen vollständigen Schutz davor geben. Dieses Problem betrifft freie und kommerzielle Software gleichermaßen. Jedoch sind Programme, die besondere Privilegien benötigen, naturgemäß gefährlicher als andere, weil jedes Schlupfloch dramatische Konsequenzen nach sich ziehen kann.(5)

Sie können niemals ausschließen, daß Ihre Sicherheitsvorkehrungen nicht doch fehlschlagen, egal, wie sorgfältig Sie waren. Sie sollten daher sicherstellen, daß Sie Eindringlinge frühzeitig erkennen. Die Überprüfung der Logdateien ist ein guter Ansatzpunkt, aber der Eindringling ist möglicherweise so clever und entfernt alle Spuren. Andererseits existieren Tools wie *tripwire*, geschrieben von Gene Kim und Gene Spafford, mit denen Sie überprüfen können, ob bei lebenswichtigen Systemdateien der Inhalt oder die Zugriffsrechte verändert wurden. *tripwire* berechnet verschiedene Checksummen über diese Dateien und speichert diese in einer Datenbank. Bei nachfolgenden Läufen werden die Checksummen erneut berechnet und mit den gespeicherten verglichen, um Veränderungen zu erkennen.

Fußnoten

(1)

Dessen ursprünglicher Geist (siehe oben) ist auch heute noch manchmal erkenntlich.

(2)

Wenn Sie bash, die GNU Bourne Again Shell, verwenden, müssen Sie dem Ausrufezeichen einen Backslash »\« voranstellen, weil es als History-Zeichen eingesetzt wird.

(3)

NetBIOS ist das Protokoll, auf dem Anwendungen wie lanmanager und Windows for Workgroups basieren.

(4)

Wir kommen darauf noch im Kapitel 9, Wichtige NetzwerkFeatures zurück.

(5)

Wenn Sie einen Netzwerkdienst anbieten, der ein setuid-Programm benötigt, müssen Sie doppelt vorsichtig sein, nichts in der Dokumentation zu übersehen, damit Sie nicht versehentlich ein Sicherheitsloch öffnen. Im Jahr 1988 brachte der RTM-Wurm einen großen Teil des Internet zum Teil dadurch zum Erliegen, daß er eine Lücke in einigen sendmail-Programmen ausnutzte. Diese Lücke wurde schon seit langen geschlossen.

Kapitel 2
Aspekte der Netzwerarbeit mit TCP/IP

Wir wenden uns nun den Details zu, mit denen Sie es beim Anschluß Ihrer Linux-Maschine an ein TCP/IP-Netzwerk zu tun bekommen, darunter auch der Umgang mit IP-Adressen, Hostnamen und manchmal auch Routing-Aufgaben. Dieses Kapitel gibt Ihnen das erforderliche Hintergrundwissen an die Hand, das Sie benötigen, um die Anforderungen Ihres Setups zu verstehen. Die nächsten Kapitel behandeln die Tools, die Sie dabei verwenden werden.

Wenn Sie mehr über TCP/IP und dessen Hintergründe erfahren wollen, sollten Sie sich das aus drei Bänden bestehende *Internetworking with TCP/IP* von Douglas R. Comer ansehen. Eine Anleitung zur Verwaltung eines TCP/IP-Netzwerks finden Sie in *TCP/IP Netzwerk Administration* von Craig Hunt.

Netzwerk-Schnittstellen

Um die ganze Vielfalt an Ausrüstung zu verstecken, die in einem Netzwerk eingesetzt werden kann, definiert TCP/IP eine abstrakte *Schnittstelle*, über die auf die Hardware zugegriffen wird. Dieses Interface bietet eine Reihe von Operationen an, die für jede Hardware gleich sind und sich im wesentlichen mit dem Senden und Empfangen von Paketen beschäftigen.

Für jedes periphere Gerät, das Sie für das Netzwerk verwenden wollen, muß ein entsprechendes Interface im Kernel präsent sein. Zum Beispiel werden Ethernet-Interfaces unter Linux *eth0* und *eth1* genannt; SLIP-Schnittstellen heißen *sl0*, *sl1* etc. Die Namen dieser Schnittstellen werden zu Konfigurationszwecken verwendet, wenn Sie eine bestimmte physikalische Einheit für den Kernel benennen wollen. Darüber hinaus besitzen sie keine Bedeutung.

Um in einem TCP/IP-Netzwerk eingesetzt werden zu können, muß dem Interface eine IP-Adresse zugewiesen werden, die als Identifizierung dient, wenn mit dem Rest der Welt kommuniziert wird. Diese Adresse ist nicht mit dem oben erwähnten Interface-Namen identisch. Vergleichen Sie ein solches Interface mit einer Tür, dann gleicht die Adresse dem an ihr befestigten Namensschild.

Natürlich können noch weitere Parameter des Geräts eingestellt werden. Einer dieser Parameter ist die maximale Größe von Datagrammen, die von der Hardware verarbeitet werden können. Diese Größe wird auch als *Maximum Transfer Unit* oder MTU bezeichnet. Andere Attribute werden später eingeführt.

IP-Adressen

Wie bereits im vorhergehenden Kapitel erwähnt, bestehen die vom IP-Netzwerk-Protokoll verstandenen Adressen aus 32-Bit-Zahlen. Jeder Maschine muß eine für die Netzwerkumgebung eindeutige Adresse zugewiesen werden. Wenn Sie ein lokales Netzwerk betreiben, das keine Verbindung zu anderen Netzwerken unterhält, dann können Sie diese Adressen nach Ihren Wünschen frei vergeben. Für Sites im Internet werden Nummern dagegen von einer zentralen Stelle, dem Network Information Center, oder kurz NIC, vergeben.(1)

Zur einfacheren Lesbarkeit werden IP-Adressen in vier 8-Bit-Zahlen aufgeteilt, die man *Oktette* nennt. Beispielsweise hat **quark.physics.groucho.edu** die IP-Adresse **0x954C0C04**, was als **149.76.12.4** geschrieben wird. Diese Schreibweise wird häufig als *Dotted Quad Notation* bezeichnet.

Ein weiterer Grund für diese Notation ist, daß sich IP-Adressen aus einer *Netzwerk*nummer in den führenden Oktets und aus einer *Host*nummer dahinter zusammensetzen. Wenn Sie beim NIC IP-Adressen beantragen, wird Ihnen nicht einfach eine bestimmte Adresse für jeden einzelnen Host zugewiesen, den Sie einbinden wollen, sondern Sie erhalten eine Netzwerknummer und dürfen innerhalb dieser Grenzen jede gültige IP-Adresse verwenden.

Abhängig von der Größe des Netzwerks muß der Adressenanteil des Host mal größer oder kleiner sein. Um diesen unterschiedlichen Bedürfnissen entgegenzukommen, gibt es unterschiedliche Netzwerk-Klassen, die unterschiedliche Aufteilungen von IP-Adressen definieren.

Klasse A

Klasse A umfaßt Netzwerke von **1.0.0.0** bis **127.0.0.0**. Die Netzwerknummer ist im ersten Oktet enthalten. Es steht also ein 24 Bit langer Hostanteil zur Verfügung, was für ca. 1,6 Millionen Hosts ausreicht.

Klasse B

Klasse B umfaßt die Netzwerke **128.0.0.0** bis **191.255.0.0**. Die Netzwerknummer ist in den ersten beiden Oktets enthalten. Das erlaubt 16.320 Netze mit jeweils 65.024 Hosts.

Klasse C

Klasse C umfaßt die Netzwerke **192.0.0.0** bis **223.255.255.0**. Die Netzwerknummer steht in den ersten drei Oktets. Das gestattet fast 2 Millionen Netzwerke mit bis zu 254 Hosts.

Klassen D, E und F

Adressen, die im Bereich von **224.0.0.0** bis **254.0.0.0** liegen, sind entweder experimenteller Natur oder für zukünftige Verwendungen reserviert und spezifizieren kein Netzwerk.

Wenn Sie zu unserem Beispiel im vorigen Kapitel zurückkehren, sehen Sie, daß **149.76.12.4**, die Adresse von **quark**, auf den Hostrechner **12.4** im Klasse-B-Netzwerk **149.76.0.0** zeigt.

Sie werden bemerkt haben, daß in der obigen Liste nicht alle möglichen Werte für jedes Oktet des Hostanteils verwendet werden dürfen. Der Grund dafür liegt darin, daß Hostnummern, bei denen alle Oktets **0** oder **255** lauten, für spezielle Zwecke reserviert sind. Eine Adresse, bei der der Hostanteil nur aus Nullen besteht, steht für das Netzwerk. Sind alle Bits des Hostanteils auf 1 gesetzt, spricht man von einer Broadcast-Adresse. Daten, die an diese Adresse geschickt werden, werden gleichzeitig von allen Hosts in dem spezifizierten Netzwerk empfangen. **149.76.255.255** ist also keine gültige Host-Adresse, sondern steht für alle Hosts im Netzwerk **149.76.0.0**.

Es existieren auch zwei reservierte Netzwerk-Adressen: **0.0.0.0** und **127.0.0.0**. Die erste ist die sogenannte *Default Route*, die zweite ist die *Loopback-Adresse*. Die Default Route hat etwas damit zu tun, wie IP Datagramme routet. Dieses Thema wird im nächsten Abschnitt behandelt.

Das Netzwerk **127.0.0.0** ist für den IP-Verkehr reserviert, der lokal auf Ihrem Host stattfindet. Üblicherweise wird die Adresse **127.0.0.1** einem speziellen Interface auf Ihrem Host, dem sogenannten *Loopback-Interface*, zugewiesen, das wie ein geschlossener Kreis funktioniert. Jedes von TCP oder UDP dorthin übergebene Paket wird direkt von dort zurückgegeben, als wäre es gerade von einem anderen Netzwerk angekommen. Auf diese Weise können Sie Netzwerk-Software entwickeln und austesten, ohne jemals ein »echtes« Netzwerk verwenden zu müssen. Eine andere sinnvolle Anwendung ergibt sich beim Einsatz von Netzwerk-Software auf einem Host, der nicht vernetzt ist. Das ist gar nicht so ungewöhnlich, wie es klingt. Beispielsweise verfügen viele UUCP-Sites gar nicht über IP-Connectivity, wollen aber trotzdem das INN-News-System einsetzen. Für den korrekten Betrieb unter Linux benötigt INN das Loopback-Interface.

Auflösung von IP-Adressen

Nachdem Sie nun gesehen haben, wie IP-Adressen aufgebaut sind, werden Sie sich vielleicht fragen, wie diese Adressen in einem Ethernet benutzt werden, um unterschiedliche Hosts anzusprechen. Schließlich verwendet das Ethernet-Protokoll ja eine 6-Byte-Zahl, um einen Host zu identifizieren, die absolut nichts mit der IP-Adresse gemeinsam hat, nicht wahr?

Richtig. Darum benötigen wir auch einen Mechanismus, der IP-Adressen auf Ethernet-Adressen abbildet. Diesen Mechanismus stellt das sogenannte *Address Resolution Protocol*, kurz ARP. Tatsächlich ist ARP nicht allein auf Ethernets beschränkt, sondern wird auch bei anderen Netzwerkarten wie Ham-Radio verwendet. Die ARP zugrundeliegende Idee

entspricht genau dem, was die meisten Leute tun, wenn sie einen ihnen unbekannten Menschen, nennen wir ihn einfach mal Karl Kiwi, in einer Gruppe von 150 Menschen herausfinden müssen: Sie gehen herum und rufen den Namen aus, in der Hoffnung, daß er sich meldet.

Wenn ARP die Ethernet-Adresse ermitteln möchte, die zu einer IP-Adresse gehört, verwendet es eine Ethernet-Eigenschaft, die als »Broadcasting,« bekannt ist. Dabei wird ein Datagramm gleichzeitig an alle Stationen im Netzwerk geschickt. Das von ARP gesendete Broadcast-Datagramm enthält eine Anfrage zur IP-Adresse. Jeder empfangende Host überprüft daraufhin seine eigene Adresse. Stimmt diese mit der empfangenen Adresse überein, wird eine ARP-Antwort an den anfragenden Host zurückgeschickt. Der empfangende Host kann nun aus dieser Antwort die Ethernet-Adresse des Senders extrahieren.

Sie werden sich nun vielleicht wundern, wie ein Host eine Internet-Adresse ansprechen kann, die sich in einem anderen Ethernet am Ende der Welt befindet bzw. woher der Host überhaupt weiß, daß die Adresse in einem Ethernet liegt. All diese Fragen führen uns zum sogenannten Routing, also dem Ermitteln des physikalischen Standorts eines Host im Netzwerk. Das ist das Thema des folgenden Abschnitts.

Lassen Sie uns aber zuvor noch etwas mehr über ARP reden. Hat ein Host erst einmal eine Ethernet-Adresse herausgefunden, speichert er diese im ARP-Cache. Auf diese Weise muß keine erneute Anfrage gestartet werden, wenn beim nächsten Mal ein Datagramm an den fraglichen Host geschickt werden soll. Nun wäre es aber nicht besonders klug, diese Information für immer zu speichern. Zum Beispiel könnte die Ethernet-Karte des anderen Host aufgrund technischer Probleme ausgetauscht werden müssen, und somit wäre der ARP-Eintrag ungültig. Darum werden die Einträge im ARP-Cache nach einiger Zeit gelöscht, um eine erneute Anfrage nach der IP-Adresse zu erzwingen.

Manchmal ist es auch notwendig, die IP-Adresse zu einer bestimmten Ethernet-Adresse zu ermitteln. Das passiert beispielsweise, wenn ein Rechner ohne eigene Fetsplatte von einem Server über das Netzwerk gebootet werden soll, eine sehr häufige Situation in lokalen Netzwerken. Ein solcher Client verfügt über nahezu keine Informationen über sich selbst -- mit Ausnahme seiner Ethernet-Adresse! Also sendet er eine Broadcast-Nachricht mit der Bitte an alle Boot-Server, ihm seine IP-Adresse mitzuteilen. Dazu existiert ein weiteres Protokoll namens *Reverse Address Resolution Protocol*, kurz RARP. Zusammen mit dem BOOTP-Protokoll dient es dem Booten von Clients ohne Laufwerk über das Netz.

IP-Routing

Wir greifen nun noch einmal die Frage auf, wie der Host, an den Pakete geschickt werden sollen, anhand der IP-Adresse gefunden wird. Verschiedene Teile der Adresse werden auf unterschiedliche Weise gehandhabt; es ist Ihre Aufgabe, entsprechende Dateien einzurichten, die angeben, wie die unterschiedlichen Teile zu handhaben sind.

IP-Netzwerke

Wenn Sie einen Brief an jemanden schreiben, versehen Sie den Briefumschlag üblicherweise mit der kompletten Adresse (Land, Postleitzahl, Straße etc.). Dann werfen Sie ihn in den Briefkasten, und die Post liefert ihn ans Ziel, d. h. er wird in das angegebene Land geschickt, wo die nationale Post ihn an die angegebene Adresse liefert. Der Vorteil dieses hierarchischen Schemas ist offensichtlich: An welchem Ort auch immer Sie den Brief aufgeben, der lokale Postangestellte weiß ungefähr, in welche Richtung er den Brief weiterleiten soll, muß sich aber nicht darum kümmern, welchen Weg der Brief nimmt, wenn dieser erstmal das Zielland erreicht hat.

IP-Netzwerke sind auf dieselbe Art und Weise strukturiert. Das gesamte Internet besteht aus einer Reihe von Netzwerken, die als *autonome Systeme* bezeichnet werden. Jedes System führt das Routing zwischen den teilnehmenden Hosts intern durch, d. h. die Aufgabe des Auslieferns eines Datagramms wird auf die Suche nach einem Pfad zum Netzwerk des Zielhost reduziert. Das bedeutet, daß, sobald ein Datagramm an *einen beliebigen* Host innerhalb eines Netzwerks übergeben wird, die weitere Verarbeitung ausschließlich von diesem Netz übernommen wird.

Subnetze

Diese Struktur spiegelt sich darin wieder, daß IP-Adressen in einen Host- und einen Netzwerkteil aufgeteilt werden. Per Standardeinstellung wird das Zielnetzwerk aus dem Netzwerkteil der IP-Adresse gewonnen. Hosts mit identischen IP-Netzwerknummern sollten innerhalb desselben Netzwerks zu finden sein, und alle Hosts innerhalb eines Netzwerks sollten dieselbe Netzwerknummer besitzen.(2)

Es macht Sinn, *innerhalb* des Netzwerks dasselbe Schema zu verwenden, weil es selbst wieder aus hunderten kleinerer Netzwerke bestehen kann, bei denen die kleinsten Einheiten physikalische Netzwerke wie Ethernets sind. Aus diesem Grund erlaubt IP die Unterteilung eines IP-Netzwerks in mehrere Unter- oder *Subnetze*.

Ein Subnetz übernimmt die Verantwortung für die Übertragung von Datagrammen innerhalb eines bestimmten Bereichs von IP-Adressen des IP-Netzes, dem es angehört. Genau wie bei den Klassen A, B und C wird es über den Netzwerkteil der IP-Adressen identifiziert. Allerdings wird der Netzwerkteil etwas erweitert und enthält nun einige Bits aus dem Host-Teil. Die Anzahl der Bits, die als Subnetznummer interpretiert werden, wird durch die sogenannte *Subnetzmaske*, oder kurz *Netmask*, bestimmt. Diese ist ebenfalls eine 32-Bit-Zahl, die die Bitmaske für den Netzwerkteil der IP-Adresse festlegt.

Das Netzwerk an der Groucho-Marx-Universität ist ein Beispiel für ein solches Netzwerk. Es verwendet die Klasse-B-Netzwerknummer **149.76.0.0**, und die Netmask lautet entsprechend **255.255.0.0**.

Intern besteht das Universitätsnetz aus verschiedenen kleineren Netzwerken, wie beispielsweise den LANs der verschiedenen Institute. Also werden die IP-Adressen noch einmal in 254 Subnetze, von **149.76.1.0** bis **149.76.254.0**, unterteilt. Dem Institut für theoretische Physik wurde beispielsweise die Nummer **149.76.12.0** zugeteilt. Der Universitäts-Backbone ist ein eigenständiges Netzwerk und besitzt die Nummer **149.76.1.0**. Diese Subnetze verwenden dieselbe IP-Netzwerknummer, während gleichzeitig das dritte Oktet verwendet wird, um sie unterscheiden zu können. Sie verwenden also die Subnetzmaske **255.255.255.0**.

Abbildung 2--1 zeigt, wie **149.76.12.4**, die Adresse von **quark**, verschieden interpretiert wird, wenn ein normales Klasse-B-Netzwerk vorliegt bzw. Subnetze verwendet werden.

Abbildung 2-1. Subnetz bei Klasse-B-Netzwerk

Es ist wichtig, festzuhalten, daß »Subnetting« (so wird die Technik des Aufbaus von Subnetzen genannt) nur eine *interne Aufteilung* des Netzwerks ist. Subnetze werden vom Besitzer (oder Administrator) des Netzwerks aufgebaut. Häufig werden Subnetze aufgebaut, um bestehende Grenzen widerzuspiegeln. Diese Grenzen können physikalischer (zwei Ethernets), administrativer (zwei Institute) oder geographischer Natur sein. Die Verantwortung für ein Subnetz wird an eine Kontaktperson delegiert. Diese Struktur wirkt sich allerdings nur auf das interne Verhalten des Netzwerks aus und ist nach außen hin völlig unsichtbar.

Gateways

Subnetting ist nicht nur organisatorisch von Vorteil, es ist häufig auch die natürliche Konsequenz aus bestehenden Hardwaregrenzen. Die Sichtweise eines Hosts in einem bestehenden Netzwerk, beispielsweise einem Ethernet, ist sehr beschränkt: Die einzigen Hosts, mit denen er sich unterhalten kann, sind die Hosts, die sich in dem gleichen Netz befinden wie er selbst. Alle anderen Hosts können nur über sogenannte *Gateways* erreicht werden. Ein Gateway ist ein Host, der an zwei oder mehr physikalische Netze gleichzeitig angeschlossen ist. Er ist so konfiguriert, daß er Pakete zwischen diesen Netzen hin und her übertragen kann.

Damit IP einfach unterscheiden kann, ob ein Host in einem lokalen physikalischen Netzwerk präsent ist, müssen verschiedene physikalische Netzwerke verschiedenen IP-Netzwerken angehören. Beispielsweise ist die Netzwerknummer **149.76.4.0** für die Hosts des LANs des mathematischen Instituts reserviert. Wird ein Datagramm an **quark** geschickt, erkennt die Netzwerk-Software auf **erdos** sofort anhand der IP-Adresse **149.76.12.4**, daß der Zielhost sich in einem anderen physikalischen Netzwerk befindet und daher nur durch ein Gateway erreicht werden kann (**sophus** per Standardeinstellung).

sophus selbst ist mit zwei verschiedenen Subnetzen verbunden: dem des mathematischen Instituts und dem des Universitäts-Backbones. Jedes wird über ein eigenes Interface (*eth0* bzw. *fddi0*) erreicht. Nun, welche IP-Adresse sollen wir ihm zuweisen? Sollen wir ihm eine auf dem Subnetz **149.76.1.0** oder auf **149.76.4.0** zuweisen?

Die Antwort lautet: auf beiden. Wird ein Host im Mathe-LAN angesprochen, soll **sophus** die IP-Adresse **149.76.4.1** verwenden, wird ein Host auf dem Backbone angesprochen, soll **149.76.1.4** verwendet werden.

Einem Gateway wird also auf jedem Netzwerk, an dem es hängt, eine IP-Adresse zugewiesen. Diese Adressen -- zusammen mit der entsprechenden Netmask -- ergeben gemeinsam die Schnittstelle, über die auf das Subnetz zugegriffen wird. Die Abbildung von Schnittstellen auf Adressen für **sophus** würde also wie in der Tabelle auf der folgenden Seite aussehen.

Interface	Adresse	Netmask
eth0	**149.76.4.1**	**255.255.255.0**
fddi0	**149.76.1.4**	**255.255.255.0**
lo	**127.0.0.1**	**255.0.0.0**

Der letzte Eintrag beschreibt das Loopback-Interface *lo*, das bereits vorgestellt wurde.

Abbildung 2--2 zeigt einen Ausschnitt der Netzwerk-Topologie an der Groucho-Marx-Universität (GMU). Hosts, die auf zwei Subnetzen zur gleichen Zeit präsent sind, sind mit beiden Adressen aufgeführt.

Abbildung 2-2. Ein Ausschnitt der Netzwerk-Topologie an der Groucho-Marx-Universität

Im allgemeinen können Sie den feinen Unterschied zwischen der Zuweisung einer Adresse an einen Host oder an sein Interface ignorieren. Bei Hosts wie **erdos**, die nur zu einem Netz gehören, spricht man im allgemeinen immer von dem Host mit dieser oder jener IP-Adresse, obwohl es genaugenommen die Ethernet-Schnittstelle ist, der diese IP-Adresse zugewiesen wurde. Andererseits ist diese Unterscheidung nur bei Gateways wirklich von Bedeutung.

Die Routing-Tabelle

Wir richten nun unsere Aufmerksamkeit darauf, wie IP ein Gateway auswählt, wenn es ein Datagramm an ein entferntes Netzwerk ausliefert.

Wir haben bereits gesehen, daß **erdos**, wenn es ein Datagramm für **quark** erhält, die Zieladresse überprüft und entdeckt, daß sich diese nicht im lokalen Netz befindet. Aus diesem Grund sendet **erdos** das Datagramm an das Default-Gateway (**sophus**), das sich nun vor die gleiche Aufgabe gestellt sieht. **sophus** erkennt, daß sich **quark** nicht in einem der Netzwerke befindet, mit denen es direkt verbunden ist. Es muß also ein weiteres Gateway finden, an das es das Datagramm weiterleiten kann. Die richtige Wahl wäre niels, das Gateway des physikalischen Instituts. Daher benötigt **sophus** einige Informationen, um zu wissen, welche Zielnetze über welche Gateways zu erreichen sind.

Die Routing-Informationen, die IP dabei verwendet, sind in einer Tabelle zu finden, die Netzwerke mit Gateways verbindet, über die sie zu erreichen sind. Ein sog. Catch-All-Eintrag (die *Default Route*) muß ebenfalls vorhanden sein. Dieses Gateway wird mit dem Netzwerk **0.0.0.0** assoziiert. Alle Pakete, die an ein unbekanntes Netzwerk gehen, werden über dieses Gateway verschickt. Auf **sophus** würde die Tabelle also wie folgt aussehen:

Netzwerk	Gateway	Interface
149.76.1.0	**-**	*fddi0*
149.76.2.0	**149.76.1.2**	*fddi0*
149.76.3.0	**149.76.1.3**	*fddi0*
149.76.4.0	**-**	*eth0*
149.76.5.0	**149.76.1.5**	*fddi0*
…	**…**	…
0.0.0.0	**149.76.1.2**	*fddi0*

Routen zu einem Netzwerk, mit dem **sophus** direkt verbunden ist, benötigen kein Gateway. In der Gateway-Spalte steht an dieser Stelle ein Bindestrich.

Routing-Tabellen können auf unterschiedliche Weise aufgebaut werden. Bei kleineren LANs ist es wahrscheinlich am effektivsten, die Tabelle von Hand zu erstellen und sie dann während der Boot-Phase mit Hilfe des *route*-Befehls (siehe *Die ARP-Tabelle*) an IP zu übergeben. Bei größeren Netzwerken werden sie zur Laufzeit von sog. *Routing-Dämonen* erzeugt und eingerichtet. Diese Dämonen laufen auf zentralen Hosts des Netzwerks und tauschen untereinander Routing-Informationen aus, um die »optimalen« Routen zwischen den teilnehmenden Netzwerken zu berechnen.

Abhängig von der Größe des Netzwerks werden verschiedene Routing-Protokolle verwendet. Für das Routing innerhalb autonomer Systeme (wie der Groucho-Marx-Universität) werden die *internen Routing-Protokolle* verwendet. Das bekannteste dieser Protokolle ist RIP, oder *Routing Information Protocol*, das im BSD *routed*-Daemon implementiert ist. Das Routing zwischen autonomen Systemen muß durch *externe Routing-Protokolle* wie EGP (*External Gateway Protocol*) oder BGP (*Border Gateway Protocol*) erledigt werden. Diese wurden (ebenso wie RIP) im *gated*-Daemon(3) der University of Cornell implementiert.

Metrische Werte

Beim auf RIP basierenden dynamischen Routing wird der beste Weg zu einem Zielhost oder -Netzwerk anhand der Anzahl der »Hops,« gewählt, d. h. anhand der Zahl von Gateways, die ein Datagramm passieren muß, bevor es sein Ziel erreicht hat. Je kürzer die Route ist, desto besser wird sie von RIP bewertet. Sehr lange Routen von 16 oder mehr Hops werden als unbrauchbar betrachtet und verworfen.

Soll RIP verwendet werden, um die für Ihr lokales Netzwerk anfallenden Routing-Informationen zu verwalten, muß auf allen Hosts *gated* laufen. Während des Bootens prüft *gated* alle aktiven Netzwerk-Interfaces. Existiert mehr als eine aktive Schnittstelle (das Loopback-Interface nicht mitgezählt), nimmt es an, daß der Host als Gateway fungiert, und *gated* verteilt aktiv Routing-Informationen. Anderenfalls verhält es sich passiv und empfängt nur eingehende RIP-Updates, um die lokale Routing-Tabelle zu aktualisieren.

Wenn *gated* die Daten aus der lokalen Routing-Tabelle verteilt, berechnet es die Länge der Route anhand einer sogenannten Metrik, die Teil des Eintrags in der Routing-Tabelle ist. Dieser Wert wird vom Systemadministrator während der Konfiguration der Route eingetragen und sollte die Kosten bezüglich der Verwendung dieser Route wiederspiegeln. Die Metrik einer Route zu einem Subnetz, mit dem der Host direkt verbunden ist, sollte also immer null sein, während ein Weg, der über zwei Gateways führt, einen Wert von zwei haben sollte. Andererseits müssen Sie sich um diese Werte keine Gedanken machen, wenn Sie *RIP* oder *gated* nicht verwenden.

Das Internet Control Message Protocol

IP hat noch ein verwandtes Protokoll, über das wir bisher noch nicht gesprochen haben. Dieses *Internet Control Message Protocol* (ICMP) genannte Protokoll wird vom Netzwerk-Kode des Kernel verwendet, um Fehlermeldungen und ähnliches an andere Hosts weiterzugeben. Nehmen wir beispielsweise an, daß Sie mal wieder vor **erdos** sitzen und über *telnet* auf Port 12345 auf **quark** zugreifen wollen. Leider existiert dort kein Prozeß, der an diesem Port horcht. Wenn das erste TCP-Paket für diesen Port bei **quark** ankommt, erkennt die Netzwerkschicht dies sofort und sendet umgehend eine ICMP-Message an **erdos** zurück, um mitzuteilen, daß der Port nicht erreichbar ist (»Port Unreachable«).

Es gibt eine ganze Reihe von Nachrichten, die ICMP versteht; viele davon beschäftigen sich mit Fehlerbedingungen. Darunter befindet sich auch eine sehr interessante, »Redirect Message« genannte Nachricht. Diese wird vom Routing-Modul erzeugt, wenn es erkennt, daß es von einem anderen Host als Gateway verwendet wird, obwohl es einen wesentlich kürzeren Weg gibt. Beispielsweise könnte die Routing-Tabelle von **sophus** nach dem Booten unvollständig sein. Die Tabelle könnte die Routen zum Mathe-Netzwerk, dem FDDI-Backbone und die Default Route enthalten, die auf das Gateway **gcc1** des Universitäts-Rechenzentrums weist. Alle Pakete, die für **quark** bestimmt sind, würden also an **gcc1** geschickt werden, und nicht an **niels**, das Gateway des physikalischen Instituts. Empfängt es ein solches Datagramm, bemerkt **gcc1**, daß dies eine schlechte Route ist und leitet das Paket an **niels** weiter, während gleichzeitig eine ICMP-Redirect-Nachricht an **sophus** gesandt wird, die dem Rechner die bessere Route mitteilt.

Nun sieht das vielleicht nach einer sehr cleveren Methode aus, mit der Sie vermeiden können, alle außer den grundlegenden Routen von Hand einzutragen. Seien Sie aber gewarnt, daß es nicht immer eine gute Idee ist, sich auf dynamische Routing-Schemata zu verlassen -- egal ob RIP oder ICMP-Redirects. ICMP-Redirect und RIP bieten Ihnen gar keine oder nur geringe Möglichkeiten, die Echtheit der empfangenen Routing-Informationen zu überprüfen. Auf diese Weise könnten bösartige Taugenichtse Ihr gesamtes Netz lahmlegen oder noch schlimmeres tun. Aus diesem Grund behandelt der Netzwerk-Kode des Kernels Redirect-Messages für Netzwerkrouten so, als wären es nur Redirects für Host-Routen.

Das Domain Name System

Auflösung von Hostnamen

Wie bereits beschrieben, dreht sich bei der Adressierung von TCP/IP-Netzwerken alles um 32-Bit-Zahlen. Es wird Ihnen aber wahrscheinlich schwerfallen, sich mehr als ein paar dieser Nummern zu merken. Aus diesem Grund sind Hosts im allgemeinen unter einem »normalen« Namen wie **gauss** oder **strange** bekannt. Es gehört zu den Pflichten der Anwendung, die zu diesem Namen gehörende IP-Adresse zu ermitteln. Dieser Prozeß wird als »Auflösung des Hostnamens« oder *Hostname Resolution* bezeichnet.

Muß ein Programm die IP-Adresse eines bestimmten Host ermitteln, ist es auf die Bibliotheksfunktionen *gethostbyname(3)* und *gethostbyaddr(3)* angewiesen. Diese und eine Reihe verwandter Funktionen werden traditionell in einer separaten Bibliothek (der sog. Resolver Library) gespeichert, unter Linux sind sie aber Teil der Standard *libc*-Bibliothek. Im allgemeinen Sprachgebrauch wird diese Sammlung von Funktionen »der Resolver« genannt.

Nun ist es bei einem kleinen Netzwerk wie einem Ethernet, ja selbst bei einem Cluster solcher Netze, nicht besonders schwer, die Tabellen zu verwalten, mit denen die Hostnamen auf Adressen abgebildet werden. Diese Informationen werden üblicherweise in einer Datei namens */etc/hosts* gespeichert. Wenn Sie einen Host hinzufügen oder entfernen, müssen Sie nur die *hosts*-Dateien auf allen Hosts entsprechend korrigieren. Das wird natürlich sehr mühsam in Netzwerken, die sich aus mehr als einer Handvoll Rechnern zusammensetzen.

Eine Lösung für dieses Problem ist das von Sun Microsystems entwickelte NIS, oder *Network Information System*, das im allgemeinen Sprachgebrauch auch YP oder *Yellow Pages* (also »Gelbe Seiten«) genannt wird. NIS speichert die *hosts*-Datei (und andere Informationen) in einer Datenbank auf einem Masterhost, vom dem Clients die Informationen im Bedarfsfall abrufen. Dennoch ist diese Lösung nur für Netzwerke mittlerer Größe, z. B. LANs, anwendbar, weil sie die Verwaltung der gesamten *hosts*-Datenbank durch eine zentrale Instanz erfordert, von der aus die Daten dann an alle Server verteilt werden müssen.

Auch im Internet wurden zu Beginn alle Adreß-Informationen in einer einzigen Datenbank namens *HOSTS.TXT* gespeichert. Diese Datei wurde vom »Network Information Center«, kurz NIC, verwaltet und mußte von allen teilnehmenden Sites heruntergeladen und installiert werden. Als das Netzwerk wuchs, traten verschiedene Probleme mit diesem Schema auf. Neben dem administrativen Aufwand, der mit der regelmäßigen Installation von *HOSTS.TXT* verbunden war, wurde die Last auf den diese Datei verteilenden Servern zu hoch. Noch schwerwiegender war das Problem, daß alle Namen beim NIC registriert werden mußten, um sicherzustellen, daß Namen nicht doppelt verwendet wurden.

Aus diesen Gründen wurde im Jahr 1984 ein neues Schema für die Auflösung von Adreß-Namen eingeführt, das sogenannte *Domain Name System*. DNS wurde von Paul Mockapetris entwickelt und widmete sich beiden Problemen. Es wird ausführlich in *DNS and BIND* von Paul Albitz und Cricket Liu beschrieben. Die eigentliche Definition findet sich in den RFCs 1033, 1034 und 1035.

Einstieg in DNS

DNS organisiert Hostnamen in einer Hierarchie von Domänen (engl. Domains). Eine Domain ist eine Sammlung von Sites, die in irgendeiner Weise zusammengehören -- sei es, daß sie ein Netzwerk bilden (z. B. alle Maschinen der Universität oder alle Hosts im BITNET), weil sie einer bestimmten Organisation (z. B. der Regierung) angehören, oder einfach, weil sie geographisch nah beieinanderliegen. Beispielsweise werden in den USA Universitäten in der Domain **edu** zusammengefaßt, wobei jede Universität bzw. jedes College nochmal unter einer Unterdomain (*Subdomain*) zusammengefaßt wird. Die Groucho-Marx-Universität könnte die Domain **groucho.edu** bekommen, wobei dem mathematischen Institut der Name

maths.groucho.edu zugewiesen würde. Bei Hosts im Netzwerk des Instituts würde der Domainname einfach an den Hostnamen angehängt, d. h. **erdos** wäre als **erdos.maths.groucho.edu** bekannt. Solch ein Name wird als »voll qualifizierter Domainname« (*fully qualified domain name*, kurz FQDN) bezeichnet, weil er den Host weltweit eindeutig identifiziert.

Abbildung 2-3. *Ein Ausschnitt des Domain-Namensraums*

Abbildung 2--3 zeigt einen Ausschnitt aus dem Namensraum (Name Space). Der Eintrag an der Wurzel dieses Baums, der durch einen einzelnen Punkt symbolisiert wird, wird passenderweise *Root Domain* genannt und umfaßt alle anderen Domains. Um zu verdeutlichen, daß ein gegebener Hostname ein voll qualifizierter Domainname, und nicht nur ein in einem relativen Verhältnis zu irgendeiner lokalen Domain stehender Name ist, wird ihm manchmal ein Punkt angehängt. Das verdeutlicht, daß der letzte Teil des Namens die Root-Domain bedeutet.

Abhängig von ihrer Position in der Namenshierarchie wird eine Domain als Top-Level, Second-Level oder Third-Level bezeichnet. Weitere Unterteilungen können zwar vorkommen, sind aber selten. Verschiedenen Top-Level-Domains werden Sie häufiger begegnen:

edu

(Meist U.S.-amerikanische) Bildungseinrichtungen wie Universitäten etc.

com

Kommerzielle Organisationen und Unternehmen.

org

Nichtkommerzielle Organisationen. Private UUCP-Netzwerke sind häufig in dieser Domain zu finden.

net

Gateways und andere administrative Hosts in einem Netzwerk.

mil

Militärische Einrichtungen der US-Armee.

gov

US-amerikanische Regierungsbehörden.

uucp

Diese Domain enthält offiziell die Namen aller UUCP-Sites, die keinen vollständigen Domainnamen haben. Sie wird aber kaum noch verwendet.

Technisch gesehen, gehören die vier erstgenannten Domains zum U.S.-amerikanischen Teil des Internet. Nichtdestotrotz können Sie in diesen Domänen auch Sites aus der restlichen Welt begegnen. Dies trifft vor allem auf die **net**-Domain zu. Nur **mil** und **gov** werden ausschließlich in den USA benutzt.

Außerhalb der USA besitzt jedes Land eine eigene Top-Level-Domain. Genutzt wird dabei der aus zwei Buchstaben bestehende Ländercode nach ISO-3166. Finnland nutzt beispielsweise die **fi**-Domain; **fr** wird von Frankreich genutzt, **de** von Deutschland und **au** von Australien. Unterhalb dieser Domain kann das NIC des jeweiligen Landes die Hostnamen nach Gutdünken organisieren. Zum Beispiel arbeitet Australien mit einer Second-Level-Domain, die sich an den internationalen Top-Level-Domains orientiert, also **com.au**, **edu.au** und so weiter. Andere, darunter auch Deutschland, benutzen diese zusätzliche Ebene nicht, arbeiten aber mit etwas längeren Namen, die direkt auf die Organisationen hinweisen, die eine bestimmte Domain betreiben. Beispielsweise sind Hostnamen wie **ftp.informatik.uni-erlangen.de** nicht unüblich. Schreiben Sie's der deutschen Tüchtigkeit zu.

Auf der anderen Seite deuten solche nationalen Domains nicht zwangsläufig an, daß ein Host in dieser Domain auch tatsächlich in diesem Land steht. Es bedeutet nur, daß der Host beim NIC dieses Landes registriert wurde. Ein schwedischer Hersteller könnte in Australien eine Niederlassung betreiben, aber trotzdem alle Hosts unter der Top-Level-Domain **se** registrieren.

Die Organisation von Namen in einer Hierarchie von Domains löst auch das Problem der Eindeutigkeit von Namen sehr elegant. Bei DNS muß ein Hostname nur innerhalb einer Domain eindeutig sein, und schon ist sichergestellt, daß er sich von allen anderen Hosts weltweit unterscheidet. Mehr noch, voll qualifizierte Namen sind einfacher zu merken. Alle diese Argumente sind für sich genommen schon Grund genug, eine große Domain in mehrere Subdomains aufzuteilen.

Aber DNS tut noch mehr für Sie. Es erlaubt Ihnen, die Verwaltung einer Subdomain auf deren Administratoren zu übertragen. Beispielsweise könnten die Betreiber am Groucho-Rechenzentrum eine Subdomain für jedes Institut einrichten. Die Subdomains **math** und **physics** haben Sie ja bereits kennengelernt. Wenn sich nun herausstellen sollte, daß das Netzwerk am physikalischen Institut zu groß und zu chaotisch ist, um es von außen zu verwalten (Physiker sind schließlich als ein etwas ungebärdiger Menschenschlag bekannt), könnten sie die Kontrolle über die Domain **physics.groucho.edu** einfach an die Administratoren dieses Netzwerks übergeben. Die Administratoren könnten beliebige Hostnamen verwenden und entsprechende IP-Adressen vergeben, ohne daß sich von außen jemand einmischte.

Zu diesem Zweck wird der Namensraum in *Zonen* aufgeteilt, die jeweils in der entsprechenden Domain wurzeln. Beachten Sie den feinen Unterschied zwischen einer *Zone* und einer *Domain*: Die Domain **groucho.edu** umfaßt alle Hosts der Groucho-Marx-Universität, während die Zone **groucho.edu** nur die Hosts einschließt, die direkt vom Rechenzentrum verwaltet werden, also beispielsweise die vom mathematischen Institut. Die Hosts am physikalischen Institut gehören einer anderen Zone, namentlich **physics.groucho.edu** an. In Abbildung 2--3 wird der Beginn einer Zone durch einen kleinen Kreis rechts neben dem Domainnamen markiert.

Namenssuche (Name Lookup) mit DNS

Auf den ersten Blick macht das ganze Theater mit Domains und Zonen die Auflösung von Namen zu einer furchtbar komplizierten Angelegenheit. Wenn also keine zentrale Stelle kontrolliert, welche Namen an welche Hosts vergeben wurden, wie soll dann eine einfache Anwendung das herausfinden?!

Nun kommt der wirklich geniale Teil von DNS. Wenn Sie die IP-Adresse von **erdos** ermitteln wollen, sagt Ihnen DNS, daß Sie sich an die Leute wenden sollen, die den Host verwalten, und die werden Ihnen das schon mitteilen.

Tatsächlich ist DNS eine gigantische verteilte Datenbank. Es setzt sich aus einer Menge von sogenannten Name-Servern zusammen, die Informationen über eine oder mehrere Domains liefern. Jede Zone besitzt mindestens zwei, meist mehrere Name-Server, die alle benötigten Informationen zu den Hosts in dieser Zone enthalten. Um die IP-Adresse von **erdos** zu ermitteln, müssen Sie nur den Name-Server der Zone **groucho.edu** ansprechen, der Ihnen die benötigten Daten zurückliefert.

Einfacher gesagt, als getan, werden Sie denken. Wie weiß ich, wie der Name-Server an der Groucho-Marx-Universität zu erreichen ist? Für den Fall, daß Ihr Computer nicht mit dem Orakel zur Auflösung von Adressen ausgestattet ist, hilft Ihnen DNS auch hier. Wenn Ihre Anwendung Informationen über **erdos** ermitteln muß, tritt es an den lokalen Name-Server heran, der eine sogenannte iterative Anfrage durchführt. Er beginnt mit einer Anfrage an einen Name-Server für die Root-Domain, bei dem er nach der Adresse für **erdos.maths.groucho.edu** nachfragt. Der Root-Name-Server erkennt, daß die Adresse nicht in seine Zuständigkeitszone fällt, sondern in eine unterhalb der **edu**-Domain. Er teilt Ihnen daraufhin mit, für weitere Informationen einen Name-Server für die **edu**-Zone anzusprechen, und liefert Ihnen eine Liste aller **edu**-Name-Server, zusammen mit deren Adressen gleich mit. Ihr lokaler Name-Server geht dann hin und fragt einen dieser Server, beispielsweise **a.isi.edu**. Auf ähnliche Weise wie der Root-Name-Server weiß **a.isi.edu**, daß die **groucho.edu**-Leute eine eigene Zone betreiben, und zeigt auf die entsprechenden Server. Der lokale Name-Server richtet seine Anfrage nach **erdos** nun an einen dieser Server, der erkennt, daß der Name zu seiner Zone gehört, und dieser liefert nun endlich die entsprechende IP-Adresse zurück.

Das sieht nach einer ganzen Menge zu übertragender Daten aus, nur um eine mickrige IP-Adresse in Erfahrung zu bringen, ist aber minimal, verglichen mit der Datenmenge, die übertragen werden müßte, wenn Sie immer noch mit *HOSTS.TXT* herumhantieren müßten. Andererseits bietet dieses Schema immer noch genügend Platz für Verbesserungen.

Um die Antwortzeiten bei zukünftigen Anfragen zu verringern, speichert der Name-Server die gewonnenen Informationen in seinem lokalen *Cache*. Das nächste Mal, wenn jemand in Ihrem lokalen Netz die Adresse eines Host der Domain **groucho.edu** benötigt, muß Ihr Name-Server den ganzen Prozeß nicht von vorne beginnen, sondern tritt direkt mit dem Name-Server von **groucho.edu** in Kontakt.(4)

Der Name-Server speichert diese Informationen nun aber nicht für immer, sondern rangiert sie nach einer bestimmten Zeit wieder aus. Die Zeitspanne, nach der eine Information ungültig wird, wird *Time to live*, kurz TTL genannt. Jedem Eintrag in der DNS-Datenbank wird eine solche TTL vom für die Zone verantwortlichen Administrator zugewiesen.

Domain-Name-Server

Name-Server, die alle Informationen zu den Hosts einer Zone gespeichert haben, besitzen die *Autorität* für diese Zone und werden manchmal auch als *Master-Name-Server* bezeichnet. Jede Anfrage nach einem Host in dieser Zone endet letztendlich bei einem dieser Master-Name-Server.

Um ein in sich geschlossenes Bild einer Zone zu liefern, müssen die Master-Server entsprechend gut synchronisiert sein. Dies wird erreicht, indem man einen der Server als *primären* Server einrichtet, der seine Zonen-Informationen aus entsprechenden Dateien bezieht, während man die anderen als *sekundäre* Server einrichtet, die die Zonendaten in regelmäßigen Abständen vom primären Server übertragen.

Ein Grund für die Verwendung mehrerer Name-Server liegt in der Verteilung der anfallenden Arbeit, ein anderer in der Redundanz. Fällt ein Name-Server aus irgendeinem Grund aus (z. B. durch einen Absturz oder eine Unterbrechung der Netzwerk-Verbindung), verteilen sich alle Anfragen auf die verbliebenen Server. Natürlich schützt Sie dieses Schema nicht vor Fehlfunktionen der Server (beispielsweise durch Bugs im Server-Programm selbst), die falsche Antworten auf alle DNS-Anfragen produzieren.

Natürlich ist auch ein Name-Server denkbar, der keine Autorität für eine Domain besitzt.(5) Eine solche Art Server ist dennoch nützlich, weil er trotzdem in der Lage ist, DNS-Anfragen für Anwendungen im lokalen Netz durchzuführen und die Informationen zu cachen. Er wird daher als *Caching-Only-Server* bezeichnet.

Die DNS-Datenbank

Sie haben gesehen, daß DNS nicht nur mit IP-Adressen für Hosts hantiert, sondern auch Informationen über Name-Server austauscht. In der Tat kann eine DNS-Datenbank eine ganze Reihe unterschiedlicher Einträge enthalten.

Eine einzelne Informationseinheit in einer DNS-Datenbank wird *Resource Record*, oder kurz RR genannt. Zu jedem Record gehört ein Typ, der die Art der enthaltenen Daten beschreibt, und eine Klasse, die den Netzwerktyp beschreibt, für den der Record gilt. Letzteres ist nötig, um unterschiedliche Adreß-Schemata zu unterstützen, wie beispielsweise

IP-Adressen (IN-Klasse) oder Hesiod-Adressen (verwendet vom Kerberos-System des MIT) und einige mehr. Der Prototyp eines Resource Records ist der A-Record, bei dem ein voll qualifizierter Domainname mit einer IP-Adresse assoziiert ist.

Natürlich kann ein Host mehr als einen Namen besitzen. Dabei muß aber einer dieser Namen der offizielle oder *kanonische Hostname* sein, während die anderen einfach nur Aliase sind, die auf diesen Namen zeigen. Der Unterschied liegt darin, daß der kanonische Hostname einen entsprechenden A-Record besitzt, während die anderen nur einen Record vom Typ CNAME verwenden, der auf den kanonischen Hostnamen zeigt.

Wir behandeln an dieser Stelle nicht alle Record-Typen, sondern heben sie uns für ein späteres Kapitel auf, und geben Ihnen hier nur ein kurzes Beispiel. Beispiel 2--1 zeigt einen Ausschnitt aus der Domain-Datenbank, die in die Name-Server für die Zone **physics.groucho.edu** geladen wird. *Beispiel 2-1. Ein Auszug aus der Datei named.hosts für das physikalische Institut*

```
; Autoritative Informationen zu physics.groucho.edu.
@         IN    SOA niels.physics.groucho.edu. janet.niels.physics.groucho.edu. {

                  940902           ; Seriennummer
                  360000           ; refresh
                  3600             ; retry
                  3600000          ; expire
                  3600             ; default ttl
                }
;
; Name-Server
                   IN    NS       niels
                   IN    NS       gauss.maths.groucho.edu.
gauss.maths.groucho.edu. IN A 149.76.4.23
```

```
;
; Theoretische Physik (Subnetz 12)
niels              IN    A        149.76.12.1
                   IN    A        149.76.1.12
nameserver         IN    CNAME    niels
otto.theory        IN    A        149.76.12.2
quark.theory       IN    A        149.76.12.4
down.collider      IN    A        149.76.12.5
strange.collider   IN    A        149.76.12.6
...
; Collider Lab. (Subnetz 14)
boson.collider     IN    A        149.76.14.1
muon.collider      IN    A        149.76.14.7
bogon.collider     IN    A        149.76.14.12
...
```

Neben den A- und CNAME-Records erkennen Sie am Anfang der Datei einen speziellen Record, der sich über mehrere Zeilen hinzieht. Dies ist der sogenannte SOA-Resource Record (*Start Of Authority*), der allgemeine Informationen zu der Zone enthält, für die der Server Autorität besitzt. Dazu gehört beispielsweise die Default-TTL für alle Records.

Beachten Sie, daß alle Namen in der Beispieldatei, die nicht mit einem Punkt enden, relativ zur Domain **groucho.edu** interpretiert werden. Der spezielle Name »@« der beim SOA-Record genutzt wird, ist eine Abkürzung für den Domainnamen selbst.

Sie haben oben gesehen, daß die Name-Server für die Domain **groucho.edu** irgendwie etwas über die Zone **physics** erfahren müssen, damit bei Anfragen (Queries) an die entsprechenden Name-Server verwiesen werden kann. Dies wird üblicherweise durch ein Paar von Records erreicht: dem NS-Record, der die FDQN des Servers enthält, und einem A-Record, der die Adresse für diesen Namen enthält. Diese Records halten den Namensraum zusammen und werden daher im Amerikanischen häufig als *Glue Records* bezeichnet. Sie sind die einzigen Beispiele für Records, bei denen eine übergeordnete Zone Informationen zu einer untergeordneten Zone enthält. Die auf die Name-Server für **physics.groucho.edu** zeigenden Records sind in Beispiel 2--2 aufgeführt.

Beispiel 2-2. Auszug aus der Datei named.hosts der GMU

```
;Zonen-Daten für die Zone groucho.edu.
 @             IN     SOA vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                     940701          ; Seriennummer
                     360000          ; aktualisieren
                     3600            ; erneuter Versuch
                     3600000         ; ausrangieren
                     3600            ; TTL-Standardeinstellung
              }
 ....
 ;
 ; Glue Records für die Zone physics.groucho.edu
 physics        IN     NS        niels.physics.groucho.edu.
                IN     NS        gauss.maths.groucho.edu.
 niels.physics  IN     A         149.76.12.1
 gauss.maths    IN     A         149.76.4.23
 ...
```

Reverse Lookup

Neben der Ermittlung einer IP-Adresse eines bestimmten Host ist es manchmal auch notwendig, den kanonischen Hostnamen zu bestimmen, der zu einer bestimmten Adresse gehört. Dies wird als *Reverse Mapping* bezeichnet und wird von verschiedenen Netzwerkdiensten genutzt, um die Identität eines Client zu überprüfen. Wenn Sie nur mit einer

hosts-Datei arbeiten, wird diese Datei bei einem Reverse Lookup einfach nach einem Host durchsucht, dem die fragliche IP-Adresse gehört. Mit DNS steht eine ausführliche Suche im Namensraum natürlich außer Frage. Statt dessen wurde eine spezielle Domain eingerichtet (**in-addr.arpa**), die die IP-Adressen aller Hosts in umgekehrter »Dotted Quad Notation« enthält. Beispielsweise entspricht die IP-Adresse **149.76.12.4** dem Namen **4.12.76.149.in-addr.arpa**. Der Resource Record-Typ, der diese Namen mit den entsprechenden kanonischen Hostnamen verbindet, ist PTR.

Das Einrichten einer Autoritätszone bedeutet üblicherweise auch, daß den Administratoren die vollständige Kontrolle darüber gegeben wird, wie sie Adressen an Namen zuweisen. Weil sie dabei üblicherweise mit einem oder mehreren IP-Netzwerken oder Subnetzen hantieren, kommt es zu einer Eins-auf-viele-Abbildung zwischen DNS-Zonen und IP-Netzwerken. Zum Beispiel umfaßt das physikalische Institut die Subnetze **149.76.8.0**, **149.76.12.0** und **149.76.14.0**.

Als Konsequenz daraus müssen neue Zonen in der **in-addr.arpa**-Domain zusammen mit der **physics**-Zone eingerichtet und an die Netzwerk-Administratoren des Instituts delegiert werden: **8.76.149.in-addr.arpa**, **12.76.149.in-addr.arpa** und **14.76.149.in-addr.arpa**. Anderenfalls müßte bei der Installation eines neuen Host im Collider Lab die Administration der übergeordneten Domain benachrichtigt und die neuen Adressen müßten in deren **in-addr.arpa**-Zonendatei eingetragen werden.

Die Zonendatenbank für Subnetz 12 ist in Beispiel 2--3 zu sehen. Die entsprechenden Glue Records in der Datenbank ihrer übergeordneten Zonen sind in Beispiel 2--4 abgebildet.

Beispiel 2-3. Ausschnitt aus der Datei named.rev für Netzwerk 149.76

```
; 12.76.149.in-addr.arpa-Domain.
 @        IN     SOA  niels.physics.groucho.edu. janet.niels.physics.groucho.edu. {
                      940902 360000 3600 3600000 3600
            }
 2        IN     PTR       otto.theory.physics.groucho.edu.
 4        IN     PTR       quark.theory.physics.groucho.edu.
 5        IN     PTR       down.collider.physics.groucho.edu.
 6        IN     PTR       strange.collider.physics.groucho.edu.
```

Beispiel 2-4. Ausschnitt aus der Datei named.rev für Subnetz 12

```
 ; 76.149.in-addr.arpa-Domain.
 @          IN     SOA vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                      940701 360000 3600 3600000 3600
                  }
 ...
 ; Subnetz 4: mathematisches Institut
 1.4        IN     PTR      sophus.maths.groucho.edu.
 17.4       IN     PTR      erdos.maths.groucho.edu.
 23.4       IN     PTR      gauss.maths.groucho.edu.
 ...
 ; Subnetz 12: physikalisches Institut, separate Zone
 12         IN     NS       niels.physics.groucho.edu.
            IN     NS       gauss.maths.groucho.edu.
 niels.physics.groucho.edu. IN  A 149.76.12.1
 gauss.maths.groucho.edu. IN  A   149.76.4.23
 ...
```

Eine wichtige Konsequenz des **in-addr.arpa**-Systems ist, daß Zonen nur als Obermengen von IP-Netzwerken eingerichtet werden können und, was noch wichtiger ist, daß die Netmasks dieser Netzwerke an Byte-Grenzen ausgerichtet sein müssen. Alle Subnetze an der Groucho-Marx-Universität verwenden die Netmask **255.255.255.0**, damit eine **in-addr.arpa**-Zone für jedes Subnetz eingerichtet werden kann. Wäre statt dessen die Netmask **255.255.255.128** verwendet worden, wäre die

Einrichtung von Zonen für das Subnetz **149.76.12.128** unmöglich, weil es keine Möglichkeit gibt, DNS mitzuteilen, daß die **12.76.149.in-addr.arpa**-Domain in zwei Autoritätszonen mit Hostnamen von **1** bis **127** und von **128** bis **255** aufgeteilt wurde.

Fußnoten

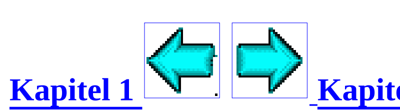
(1)
: Häufig werden Ihnen die IP-Adressen auch vom Provider zugewiesen, von dem Sie die IP-Connectivity einkaufen. Sie können sich aber auch direkt an das NIC wenden, um eine IP-Adresse für Ihr Netzwerk zu beantragen. Senden Sie dazu einfach eine Mail an hostmaster@internic.net.

(2)
: Autonome Systeme sind noch etwas allgemeiner. Sie können mehr als ein Netzwerk umfassen.

(3)
: routed wird von vielen Leuten als mangelhaft angesehen. Weil gated auch RIP unterstützt, sollten Sie lieber darauf zurückgreifen.

(4)
: Wenn die Informationen nicht in einem Cache gespeichert würden, wäre DNS genauso gut oder schlecht wie jede andere Methode, weil jede Anfrage über die Root-Name-Server laufen müßte.

(5)
: Nun, nahezu. Ein Name-Server muß zumindest den Name-Service für localhost und Reverse Lookups für 127.0.0.1 durchführen.

Inhaltsverzeichnis  Kapitel 1  Kapitel 3

Kapitel 3
Konfiguration der Hardware

Geräte, Treiber und der ganze Rest

Bis jetzt haben wir uns ziemlich ausführlich mit Netzwerk-Schnittstellen und allgemeinen Fragen der TCP/IP-Protokolle befaßt, ohne uns groß darum zu kümmern, was eigentlich genau passiert, wenn »der Netzwerk-Kode« im Kernel ein Stück Hardware anspricht. Bevor wir das jetzt nachholen, müssen wir noch ein wenig über das Konzept der Schnittstellen und Treiber sprechen.

Zuallererst ist da natürlich die Hardware selbst, zum Beispiel eine Ethernet-Karte. Das ist ein Stückchen Epoxy voller kleiner Chips mit komischen Nummern drauf, das in einem Steckplatz Ihres PC sitzt. Das ist, was wir allgemein ein Gerät nennen (engl. *device*).

Um die Ethernet-Karte benutzen zu können, müssen in Ihrem Linux-Kernel spezielle Funktionen vorhanden sein, die in der Lage sind, das Gerät anzusprechen. Diese Gruppe von Funktionen nennt man Gerätetreiber (*device driver*). Linux besitzt beispielsweise Treiber für eine ganze Reihe von Ethernet-Karten unterschiedlicher Hersteller, die einander in ihrer Funktionsweise ähneln. Sie sind als die Becker-Treiberserie bekannt, nach ihrem Autor Donald Becker benannt. Ein anderes Beispiel ist der D-Link-Treiber, der den D-Link-Taschenadapter bedient, der an den Parallelport angeschlossen wird.

Aber was meinen wir eigentlich, wenn wir sagen, ein Treiber »bediene« ein Gerät? Kommen wir noch einmal auf die oben betrachtete Ethernet-Karte zurück. Der Treiber muß in der Lage sein, mit den Logikbausteinen auf der Karte zu kommunizieren, während die Karte umgekehrt alle empfangenen Daten irgendwie beim Treiber abliefern muß.

Abbildung 3-1. Die Beziehungen zwischen Treibern, Schnittstellen und der Hardware.

In PCs findet diese Kommunikation durch einen speziellen Speicherbereich statt, der auf I/O-Register auf der Karte abgebildet wird. Alle Befehle, die der Kernel an die Karte schickt, müssen durch diese Register übermittelt werden. Dieser I/O-Speicher wird im allgemeinen mit seiner Start- oder *Basisadresse* angegeben. Typische Basisadressen für Ethernet-Karten sind `0x300` oder `0x360`.

Für gewöhnlich brauchen Sie sich über solche Details wie Basisadressen nicht den Kopf zu zerbrechen, da der Kernel beim Booten versucht, die Position der Karte selbst herauszufinden. Dieser Vorgang heißt Autoprobing, was bedeutet, daß der Kernel verschiedene Speicheradressen ausliest und mit dem vergleicht, was herauskommen sollte, wenn eine bestimmte Ethernet-Karte installiert wäre. Allerdings kann es Ethernet-Karten geben, die der Kernel nicht automatisch erkennt; das ist bei manchen nicht hundertprozentigen Kopien von Karten anderer Hersteller der Fall. Außerdem gibt sich der Kernel beim Booten schon zufrieden, wenn er nur eine Karte gefunden hat; wenn Sie mehr

als eine verwenden wollen, müssen Sie diese Karten von Hand konfigurieren.

Ein anderer Parameter, den Sie dem Kernel unter Umständen mitteilen müssen, ist der Interrupt-Kanal. Hardware-Komponenten schicken dem Kernel gewöhnlich eine Unterbrechungsanforderung, wenn sie seine Aufmerksamkeit benötigen, z. B. wenn Daten angekommen sind oder ein besonderer Zustand auftritt. In PCs können solche Unterbrechungen auf einem von 15 Interrupt-Kanälen signalisiert werden, die die Nummern 0, 1 und 3 bis 15 haben. Die einem Interrupt zugeordnete Nummer wird im Englischen *interrupt request number* oder kurz IRQ genannt.(1)

Wie in Kapitel 2, *Aspekte der Netzwerarbeit mit TCP/IP* beschrieben, spricht der Kernel ein Gerät durch eine sogenannte Schnittstelle oder Interface an. Interfaces bieten einen Satz abstrakter Funktionen, der für alle Arten von Hardware gleich ist, wie zum Beispiel das Senden oder Empfangen eines Datagramms.

Jedes Interface wird durch einen Namen bezeichnet. Diese Namen sind keine Gerätedateien im Verzeichnis */dev*, sondern werden intern vom Kernel vergeben. Typische Beispiele sind *eth0, eth1* usw. Die Zuordnung zwischen Schnittstellen und Geräten hängt gewöhnlich von der Reihenfolge ab, in der die Geräte konfiguriert werden; die erste Ethernet-Karte wird *eth0*, die nächste *eth1* und so fort. Eine Ausnahme von dieser Regel sind unter anderem die SLIP-Schnittstellen, die dynamisch zugewiesen werden; das heißt, sobald eine SLIP-Verbindung hergestellt wurde, wird dem seriellen Port ein Interface zugewiesen.

Abbildung 3--1 versucht, den Zusammenhang zwischen den Hardware-Komponenten, Gerätetreibern und Schnittstellen zu verdeutlichen.

Während des Boot-Vorgangs zeigt der Kernel an, welche Geräte er erkennt und welche Schnittstellen initialisiert werden. Das folgende ist ein Ausschnitt eines typischen Boot-Vorgangs:

```
.
.
This processor honours the WP bit even when in supervisor mode. Good.
Swansea University Computer Society NET3.017
NET3 TCP/IP protocols stack v016
Swansea University Computer Society NET3.017
Swansea University Computer Society TCP/IP for NET3.018
IP Protocols: IGMP, ICMP, UDP, TCP
PPP: version 0.2.7 (4 channels) NEW_TTY_DRIVERS OPTIMIZE_FLAGS
TCP compression code copyright 1989 Regents of the University of California
PPP line discipline registered.
SLIP: version 0.8.1-NET3.014-NEWTTY (4 channels) (6 bit encapsulation enabled)
CSLIP: code copyright 1989 Regents of the University of California
NE*000 ethercard probe at 0x340: 00 40 05 12 c6 54
eth0: NE2000 found at 0x340, using IRQ 15. ne.c:v1.10 9/23/94
Donald Becker (becker@cesdis.gsfc.nasa.gov)
Checking 386/387 coupling... Ok, fpu using exception 16 error reporting.
Checking 'hlt' instruction... Ok.
Linux version 1.1.75 (okir@monad.swb.de) (gcc version 2.6.2)
#14 Mon Jan 2 09:18:46 MET 1995 …
```

Die Meldungen zeigen, daß der Kernel netzwerkfähig ist und Treiber für SLIP, CSLIP und PPP eingebunden wurden. Die dritte Zeile von unten besagt, daß eine NE2000-kompatible Netzwerkkarte erkannt und als Interface *eth0* installiert wurde. Wenn Sie eine Ethernet-Karte installiert haben, aber keine derartige Meldung sehen, bedeutet das, daß der Kernel nicht in der Lage war, Ihre Karte zu erkennen. Dieses Problem werden wir in einem späteren Abschnitt behandeln.

Konfigurieren des Kernels

Die meisten Linux-Distributionen kommen mit Boot-Disketten einher, die mit allen gängigen Typen von PC-Hardware funktionieren. Das bedeutet, daß die Kernel auf diesen Boot-Disketten neben einer RAM-Disk auch einen ganzen Schwung Treiber enthalten, die Sie gar nicht brauchen, und die nur wertvollen Speicherplatz verschwenden. Deshalb sollten Sie auf jeden Fall einen auf Ihre Situation zugeschnittenen Kernel zusammenstellen, der nur die Treiber enthält, die Sie wirklich brauchen.

Wenn Sie ein Linux-System benutzen, sollten Sie bereits damit vertraut sein, wie man einen Kernel »baut.« Die Grundlagen dafür werden unter anderem in dem Buch »Linux -- Wegweiser zur Installation und Konfiguration« von Matt Welsh und Lar Kaufman beschrieben, weswegen wir uns hier nur mit den Optionen befassen werden, die den Netzwerkteil betreffen. Außerdem liegt den Kernel-Quellen eine README-Datei bei, die Ihnen eine »Quick and Dirty«-Einführung in die grundlegenden Dinge der Kernel-Konfiguration gibt.

Wenn Sie **`make config`** aufrufen, werden Sie zuerst nach allgemeinen Einstellungen gefragt, beispielsweise, ob Sie Treiber für IDE-Platten benötigen oder nicht. Unter anderem bietet Ihnen das Konfigurations-Skript Unterstützung für TCP/IP-Networking an. Diese Frage müssen Sie mit Ja (**`y`**) beantworten, um einen netzwerkfähigen Kernel zu erhalten.

Kerneloptionen in Linux 1.0 und höher

Nachdem Ihnen das Skript verschiedene Optionen wie a. a. SCSI-Treiber angeboten hat, folgt ein Abschnitt zur Netzwerk-Konfiguration. Eine typische Liste von Optionen, wie sie Kernel der Reihen 1.0 und 1.1 anbieten, sieht etwa so aus:

```
*
* Network device support
*
Network device support? (CONFIG_ETHERCARDS) [y]
```

Trotz des etwas irreführenden Makronamens in Klammern müssen Sie diese Frage grundsätzlich mit **`y`** beantworten, falls Sie *irgendeine* Art von Netzwerk-Treibern einbinden wollen, sei es Ethernet, SLIP oder PPP. Wenn Sie mit Ja antworten, wird zunächst die grundlegende Funktionalität für Ethernet-Karten zu Verfügung gestellt. Andere Treiber müssen ausdrücklich aktiviert werden.

```
SLIP (serial line) support? (CONFIG_SLIP) [y]  SLIP compressed headers
(SL_COMPRESSED) [y] PPP (point-to-point) support (CONFIG_PPP) [y] PLIP
(parallel port) support (CONFIG_PLIP) [n]
```

Diese Fragen betreffen die verschiedenen Protokolle, die von Linux auf der Verbindungsebene (*link layer*) bereitgestellt werden. SLIP erlaubt Ihnen, IP-Datagramme über serielle Leitungen zu transportieren. Die Option »compressed headers« stellt zusätzliche Unterstützung für CSLIP bereit, eine SLIP-Variante, die TCP/IP-Header auf bis zu drei Byte komprimieren kann. Es sei darauf hingewiesen, daß diese Kernel-Option CSLIP beim Verbindungsaufbau nicht automatisch einschaltet; sie stellt bloß die notwendige Funktionalität zur Verfügung.

PPP ist ein weiteres Protokoll, um Netzverkehr über serielle Leitungen zu übertragen. Es ist wesentlich flexibler als SLIP, da es nicht auf IP beschränkt ist. Grundsätzlich ist PPP auch in der Lage, beispielsweise IPX zu transportieren; das ist allerdings noch nicht implementiert. Da es den PPP-Treiber noch nicht allzulange gibt, fehlt diese Option in älteren Kernel-Versionen.

PLIP bietet die Möglichkeit, IP-Datagramme über parallele Verbindungen zu übertragen. Es wird häufig zur

Kommunikation mit PCs verwendet, die DOS fahren.

Die folgenden Fragen befassen sich mit Ethernet-Karten verschiedener Hersteller. Da ständig neue Treiber entwickelt werden, verändert sich die genaue Struktur dieses Abschnitts ständig. Sie sollten sich dadurch nicht irritieren lassen. Wenn Sie einen Kernel zusammenstellen wollen, der auf mehr als einer Maschine laufen soll, können Sie auch mehr als einen Treiber auswählen.

```
NE2000/NE1000 support (CONFIG_NE2000) [y]
WD80*3 support (CONFIG_WD80x3) [n]
SMC Ultra support (CONFIG_ULTRA) [n]
3c501 support (CONFIG_EL1) [n]
3c503 support (CONFIG_EL2) [n]
3c509/3c579 support (CONFIG_EL3) [n]
HP PCLAN support (CONFIG_HPLAN) [n]
AT1500 and NE2100 (LANCE and PCnet-ISA) support (CONFIG_LANCE) [n]
AT1700 support (CONFIG_AT1700) [n]
DEPCA support (CONFIG_DEPCA) [n]
D-Link DE600 pocket adaptor support (CONFIG_DE600) [y]
AT-LAN-TEC/RealTek pocket adaptor support (CONFIG_ATP) [n]
*
* CD-ROM drivers
*
```

...

Im Abschnitt über Dateisysteme fragt das Konfigurations-Skript Sie schließlich, ob Sie Unterstützung für NFS, das Netzwerk-Dateisystem (*Network File System*) wünschen. NFS erlaubt Ihnen, Dateisysteme an verschiedene Maschinen zu exportieren, so daß die Dateien auf diesen Rechnern erscheinen, als befänden sie sich auf einer ganz normalen lokalen Festplatte.

```
NFS filesystem support (CONFIG_NFS_FS) [y]
```

Kerneloptionen in Linux 1.1.14 und höher

Seit Version 1.1.14, die unter anderem Alpha-Unterstützung für IPX einführte, hat sich die Konfigurations-Prozedur geringfügig verändert. Der Teil, der sich mit allgemeinen Optionen befaßt, fragt Sie nun, ob Sie überhaupt irgendeine Art von Unterstützung für TCP/IP wünschen. Dem folgt unmittelbar eine Reihe von Fragen über verschiedene Netzwerk-Optionen.

```
*
* Networking options
* TCP/IP networking (CONFIG_INET) [y]
```

Um TCP/IP -- Networking (FIXME) zu benutzen, müssen Sie diese Frage mit Ja (**y**) beantworten. Wenn Sie mit **n** antworten, haben Sie aber immer noch die Möglichkeit, IPX-Unterstützung einzubinden.

```
IP forwarding/gatewaying (CONFIG_IP_FORWARD) [n]
```

Diese Option müssen Sie einschalten, wenn Ihr System als Gateway zwischen zwei Ethernets oder einem Ethernet und einer SLIP-Verbindung usw. dienen soll. Obwohl es nicht schadet, IP-Forwarding grundsätzlich bereitzustellen, möchten Sie vielleicht darauf verzichten, um Ihr System als sogenannten Firewall (wörtlich: Brandschutzmauer) zu konfigurieren. Firewalls sind Rechner, die zwischen zwei oder mehreren Netzen sitzen, aber keine Pakete vom einen

ins andere routen. Sie werden häufig dazu benutzt, um beispielsweise Mitarbeitern einer Firma Zugang zum Internet zu bieten, aber gleichzeitig die Risiken für das firmeninterne Netz zu begrenzen. Benutzer können sich auf dem Firewall einloggen und Internet-Dienste benutzen, aber alle anderen Maschinen sind vor Angriffen von außen geschützt, da hereinkommende Verbindungen nur bis zum Firewall kommen.(2)

```
*
* (it is safe to leave these untouched)
*
PC/TCP compatibility mode (CONFIG_INET_PCTCP) [n]
```

Diese Option umgeht eine Inkompatibilität mit einigen Versionen von PC/TCP, einer kommerziellen TCP/IP-Implementation für DOS-basierte PCs. Wenn Sie diese Option einschalten, können Sie immer noch mit normalen UNIX-Maschinen kommunizieren; allerdings kann der Durchsatz über langsame Verbindungen wie SLIP etwas darunter leiden.

```
Reverse ARP (CONFIG_INET_RARP) [n]
```

Diese Funktion stellt RARP, das *Reverse Address Resolution Protocol*, bereit. RARP bedeutet soviel wie »umgekehrte Adreßauflösung« und wird von X-Terminals und ähnlichem benutzt, um beim Booten die eigene IP-Adresse herauszufinden. Sie sollten RARP nur dann einschalten, wenn Sie diese Art von Geräten bedienen wollen. Neuere Versionen der Netzwerk-Utilities enthalten ein Programm names *rarp*, das Ihnen erlaubt, Systeme in den RARP-Cache des Kernels einzutragen.

```
Assume subnets are local (CONFIG_INET_SNARL) [y]
```

Wenn Sie Daten über eine TCP-Verbindung transportieren, muß der Kernel sie in kleinere Pakete zerteilen, bevor er sie an die IP-Schicht weiterreicht. Ist das Zielsystem über ein lokales Netz wie ein Ethernet erreichbar, werden größere Pakete benutzt als für Systeme, die nur über langsame Weitverbindungen erreichbar sind.(3) Wenn Sie `SNARL` nicht einschalten, nimmt der Kernel an, daß nur solche IP-Netze wirklich lokal sind, zu denen er ein direktes Interface hat. Betrachten Sie andererseits die Groucho-Marx-Universität: Hier sind alle Teilnetze durch einen schnellen Backbone miteinander verbunden, aber die meisten Rechner sind nur an ein oder höchstens zwei Subnetze direkt angeschlossen. Mit `SNARL` nimmt der Kernel nun an, daß *alle* Subnetze lokal sind, und verwendet immer große Pakete, wenn er mit einem anderen Rechner auf dem Campus spricht.

Wenn Sie `SNARL` wählen, aber trotzdem für einige ausgewählte Rechner kleinere Pakete verwenden wollen (beispielsweise weil die Daten über eine SLIP-Leitung gehen), können Sie die `mtu`-Option des *route*-Kommandos verwenden, das am Ende dieses Kapitels kurz angesprochen wird.

```
Disable NAGLE algorithm (normally enabled) (CONFIG_TCP_NAGLE_OFF) [n]
```

Die Regel von Nagle ist eine Heuristik, um die Erzeugung besonders kleiner IP-Pakete, auch Tinygrams genannt, zu vermeiden. Tinygrams werden unter anderem von interaktiven Programmen erzeugt, die einzelne Tastenanschläge übertragen, wie *telnet* oder *rlogin*. Tinygrams sind insbesondere auf Verbindungen wie SLIP oder PPP, die eine recht niedrige Bandbreite haben, eine ziemliche Verschwendung. Der Algorithmus von Nagle versucht sie zu vermeiden, indem er die Übertragung von TCP-Daten unter bestimmten Umständen kurzfristig verzögert. Sie sollten nur dann auf Nagles Algorithmus verzichten, wenn schwere Probleme durch Paketverluste auftreten.

```
The IPX protocol (CONFIG_IPX) [n]
```

Diese Option stellt Unterstützung für IPX, ein von Novell Networking benutztes Protokoll, bereit. Es befindet sich derzeit noch in der Entwicklung. Der Hauptnutzen von IPX wird sein, daß Sie eines Tages Daten mit IPX-basierten

DOS-Utilities austauschen und Verkehr zwischen Novell-Netzen beispielsweise über eine PPP-Verbindung routen können. Unterstützung für die höheren Novell-Protokolle wie NCP ist dagegen nicht in Sicht, da die nötigen Spezifikationen nur zu erklecklichen Preisen und gegen eine Non-Disclosure-Vereinbarung erhältlich sind.

Seit der Version 1.1.16 unterstützt Linux einen weiteren Treibertyp, den Dummy-Treiber. Die folgende Frage taucht am Anfang des Abschnitts über Gerätetreiber auf:

```
Dummy net driver support (CONFIG_DUMMY) [y]
```

Der Dummy-Treiber tut nicht besonders viel, wie der Name auch schon ahnen läßt, ist aber auf einem Rechner ohne Netzanbindung oder mit nur einer SLIP-Verbindung sehr nützlich. Er ist im wesentlichen ein maskiertes Loopback-Interface. Der Grund, eine solche Schnittstelle zu haben, ist folgender: Auf einer Maschine, die SLIP, aber kein Ethernet hat, wünscht man sich eine Schnittstelle, die die ganze Zeit Ihre IP-Adresse trägt, und nicht nur, wenn die SLIP-Verbindung aktiv ist. Wir werden auf diese Frage noch etwas ausführlicher in Abschnitt *Die Dummy-Schnittstelle* in Kapitel 5 zurückkommen.

Eine Tour durch die Netzwerk-Geräte

Der Linux-Kernel unterstützt eine Anzahl von Treibern für unterschiedliche Arten von Hardware. Dieser Abschnitt gibt Ihnen einen kurzen Überblick der vorhandenen Treiber-Familien und der für sie reservierten Interface-Namen. Die meisten Treiber unterstützen mehrere Schnittstellen. In diesem Falle werden die Namen durchnumeriert, z. B. als *eth0*, *eth1* usw.

lo

Das Loopback-Interface. Es kann einerseits zum Testen verwendet werden, wird aber auch von einigen Applikationen wie INN benutzt, um Daten darüber auszutauschen. Es funktioniert wie eine Art Schleife, die jedes empfangene Datagramm umgehend an die Netzwerkschicht Ihres Systems zurückreicht. Jeder Kernel besitzt eine solche Schnittstelle, und es ist im allgemeinen auch nicht sinnvoll, mehr oder weniger zu haben.

eth0, eth1, …

Ethernet-Geräte. Dies ist der generische Name für die meisten Ethernet-Karten.

dl0, dl1, …

Diese Schnittstellen bedienen den D-Link DE-600 Pocket-Adapter. Er unterscheidet sich von gewöhnlichen Ethernet-Geräten dadurch, daß er durch den Parallelport angesteuert wird. Ab der Kernel-Revision 1.1 ist dieser Name allerdings obsolet; der DE-600 wird jetzt auch als eth0, eth1 etc. registriert.

sl0, sl1, …

SLIP-Schnittstellen. Die Namen werden in der Reihenfolge vergeben, in der serielle Ports für die SLIP-Nutzung belegt werden, d. h. die erste serielle Verbindung, die für SLIP konfiguriert wird, erhält den Namen sl0, die nächste sl1 usf. Per Voreinstellung unterstützt der Kernel bis zu vier SLIP-Interfaces.

ppp0, ppp1, …

PPP-Schnittstellen. Sie werden wie SLIP-Kanäle auch erst dann zugeteilt, wenn ein serieller Port in den PPP-Modus geschaltet wird. Zur Zeit werden bis zu vier Interfaces unterstützt.

plip0, plip1, …

PLIP-Schnittstellen. PLIP überträgt IP-Pakete über den Parallelport. Bis zu drei PLIP-Interfaces werden unterstützt. Sie werden vom PLIP-Treiber während des Bootens reserviert und den einzelnen Parallelports statisch zugeordnet.

In Zukunft werden für andere Treiber wie ISDN oder Amateurfunk (Ham-Radio) weitere Namen eingeführt werden.

In den folgenden Abschnitten werden wir uns im einzelnen mit den oben aufgeführten Treibern beschäftigen.

Ethernet-Installation

Wie bereits mehrfach erwähnt, stellt Linux Ethernet-Treiber für Karten diverser Hersteller bereit. Die meisten Treiber stammen von Donald Becker (*becker@cesdis.gsfc.nasa.gov*), der eine Treiber-Familie für Karten geschrieben hat, die auf dem Chip NSC 8390 von National Semiconductor basieren. Daneben gibt es Treiber für einige Produkte von D-Link, unter anderem für den bereits erwähnten Pocket-Adapter. Dieser Treiber wurde von Bjørn Ekwall geschrieben (erreichbar als *bj0rn@blox.se* -- Vorsicht, der Buchstabe *0* ist eine Null). Der DEPCA-Treiber stammt von David C. Davies. In jüngster Zeit hat David für DEC auch einen Treiber für die Digital EtherWorks 3-Karte geschrieben, der von DEC unter der GNU-Lizenz freigegeben wurde.(4)

Verkabelung

Falls Sie das erste Mal in Ihrem Leben ein Ethernet installieren, sind hier einige Worte über die Verkabelung angebracht. Ethernet ist sehr pingelig, was den korrekten Anschluß der Kabel angeht. Das Kabel muß an beiden Enden mit einem 50 Ohm-Widerstand abgeschlossen werden, und es dürfen keine Verzweigungen auftreten (d. h. drei sternförmig verbundene Kabel). Wenn Sie das gängige dünne Koaxialkabel mit den T-förmigen BNC-Verbindern benutzen, müssen diese Verbinder direkt auf die Karte aufgesteckt werden; Sie sollten unter keinen Umständen ein Verlängerungskabel einfügen.

Die Länge des Kabels spielt ebenfalls eine wichtige Rolle. Bei einer Installation mit dünnem Koaxialkabel darf der Strang nicht länger als 200 Meter sein,(5) und Stecker müssen mindestens 2.5 Meter auseinander liegen. Wenn Sie einen längeren Strang benötigen, um alle Maschinen miteinander zu verbinden, müssen Sie an geeigneten Stellen sogenannte Repeater einfügen, die das Signal verstärken.

Wenn Sie sich an eine Thicknet-Installation anschließen, die das dickere Koaxialkabel (ca. 8 mm) verwendet, benötigen Sie einen Transceiver, auch *Ethernet Attachment Unit* genannt. Der Transceiver wird im allgemeinen über ein abgeschirmtes Kabel mit dem AUI-Port Ihrer Karte verbunden.

Unterstützte Karten

Die folgende Liste beschreibt die verbreitetsten Karten, die von Linux unterstützt werden. Eine vollständige Liste, die ungefähr dreimal so lang ist, finden Sie im Ethernet-HOWTO von Paul Gortmaker.(6) Die aktuellen HOWTOs sind per FTP auf **ftp.uni-erlangen.de** im Verzeichnis *MIRROR.sunsite/docs* erhältlich und sollten auch den meisten Linux-CDs beiliegen.

Selbst wenn Sie Ihre Karte in der folgenden Liste finden, schauen Sie bitte trotzdem ins HOWTO. Es diskutiert für einige Karten verschiedene Betriebsmodi, die die Leistung Ihrer Karte deutlich beeinflussen können.

3Com Etherlink

: Sowohl 3c503 als auch 3c503/16 werden unterstützt, wie auch 3c507 und 3c509. Es gibt auch einen Treiber für die 3c501, allerdings ist die Karte so langsam, daß sich die Investition kaum lohnt. Alan Cox benutzt nach eigenem Bekunden zwei 3c501-Karten, um die Fehlertoleranz des Netzwerk-Kodes zu testen.

Novell Eagle

: NE1000 und NE2000 sowie eine Reihe von Nachbauten anderer Hersteller. NE1500 und NE2100 werden ebenfalls unterstützt.

Western Digital/SMC

: WD8003 und WD8013 (diese Karten entsprechen der SMC Elite bzw. SMC Elite Plus). Daneben wird die 16-Bit-Version SMC Elite Ultra unterstützt.

Hewlett Packard

HP 27252, HP 27247B, und HP J2405A.

D-Link

DE-600 Pocket-Adapter, DE-100, DE-200, DE-220T und DE-620. Daneben existiert ein Patch-Kit für die DE-650, eine PCMCIA-Karte.(7)

Digital Equipment

Die DEPCA-Karten DEPCA rev E, DE200 (32K/64K), DE201, DE202, DE210, DE100 und DE422. Außerdem seit Linux 1.1.60 auch die Etherworks-Karten DE203, DE204 und DE205.

Um eine dieser Karten mit Linux zu verwenden, können Sie einen fertigen Kernel aus einer der vielen Linux-Distributionen verwenden. Diese Kernel haben im allgemeinen Treiber für alle Karten eingebunden. Auf lange Sicht ist es allerdings besser, wenn Sie sich Ihren eigenen Kernel bauen und nur diejenigen Treiber verwenden, die Sie tatsächlich brauchen.

Ethernet-Autoprobing

Beim Booten versucht der Kernel herauszufinden, ob und was für eine Ethernet-Karte Sie installiert haben. Dabei prüfen alle Treiber nacheinander bestimmte Adressen, an denen sich eine Karte befinden kann, und gehen dabei nach dieser Reihenfolge vor:

```
--------------------------------------------------------------------
Karte          Geprüfte Adressen
--------------------------------------------------------------------
WD/SMC         0x300, 0x280, 0x380, 0x240
SMC 16 Ultra   0x300, 0x280
3c501          0x280
3c503          0x300, 0x310, 0x330, 0x350, 0x250, 0x280, 0x2a0, 0x2e0
NEx000         0x300, 0x280, 0x320, 0x340, 0x360
HP             0x300, 0x320, 0x340, 0x280, 0x2C0, 0x200, 0x240
DEPCA          0x300, 0x320, 0x340, 0x360
--------------------------------------------------------------------
```

Der Ethernet-Testkode hat zwei Einschränkungen: Zum einen ist es möglich, daß eine Karte nicht korrekt erkannt wird. Das ist zwar recht selten, kommt aber gelegentlich bei billigeren Kopien gängiger Marken wie NE2000 vor. Einige der WD80x3-Karten sollen dieses Problem auch haben.

Die zweite Einschränkung ist, daß der Kernel sich zufriedengibt, sobald er eine Karte gefunden hat, und sich nicht weiter um eine eventuell vorhandene, zweite Karte kümmert. Das ist durchaus beabsichtigt, da davon ausgegangen wird, daß Sie die Kontrolle darüber haben wollen, welches Interface welcher Karte zugewiesen wird.

Wenn Sie zwei Karten verwenden oder wenn der Selbsttest-Kode Ihre Karte nicht erkennt, müssen Sie dem Kernel die Basisadresse und evtl. den IRQ der Karte ausdrücklich mitteilen. Das können Sie auf zwei Arten tun.

Ein gangbarer Weg ist, die Datei *drivers/net/Space.c* in den Kernel-Quellen zu ändern. Diese Datei enthält alle wesentlichen Informationen über die Netzwerk-Treiber. Das ist aber nur empfehlenswert, wenn Sie sich etwas mit dem Netzwerk-Kode des Kernels auskennen. Eine wesentlich bessere Methode ist, dem Kernel diese Information beim Booten mitzuteilen. Wenn Sie *lilo* als Boot-Lader einsetzen, können Sie dem Kernel verschiedene Parameter direkt beim Booten oder durch die `append`-Option in der Datei *lilo.conf* übergeben. Der `ether`-Parameter beschreibt ein Ethernet-Gerät und hat folgendes Format:

```
ether=irq,base_addr,param1,param2,name
```

Die ersten vier Parameter sind Zahlen, während der letzte der Name des zuzuordnenden Interfaces ist. Alle numerischen Werte sind optional; wenn sie weggelassen oder auf Null gesetzt werden, wird der Kernel den Wert automatisch herauszufinden versuchen bzw. eine Voreinstellung verwenden.

Der erste Parameter legt den IRQ des Gerätes fest. Ist kein Wert angegeben, versucht der Kernel, den IRQ-Kanal selbst festzustellen. Der 3c503-Treiber hat ein spezielles Feature, das einen freien IRQ aus der Liste 5, 9, 3 und 4 auswählt und die Karte so konfiguriert, daß sie diesen Interrupt verwendet.

Der Parameter *`base_addr`* gibt die I/O-Adresse der Karte an; ein Wert von Null veranlaßt den Kernel, sie über den üblichen Autoprobing-Mechanismus zu suchen.

Die verbleibenden zwei Parameter werden von verschiedenen Treibern unterschiedlich interpretiert. Für Karten wie die WD80x3, die auf Shared Memory basieren, geben Sie die Anfangs- und Endadresse des Shared Memory-Bereichs an. Die meisten anderen Karten interpretieren *`param1`* als den Umfang, in dem Debugging-Informationen ausgegeben werden. Werte von 1 bis 7 bezeichnen zunehmende Geschwätzigkeit, während 8 diese Informationen völlig unterdrückt. Null wählt einen Voreinstellungswert aus, der von Karte zu Karte unterschiedlich sein kann. Der 3c503-Treiber verwendet *`param2`*, um zwischen dem internen Transceiver (Voreinstellung) und einem externen Transceiver (Wert von 1) zu unterscheiden. Die erste Variante stellt die BNC-Buchse der Karte ein, die zweite den AUI-Port.

Wenn Sie zwei Karten verwenden, können Sie Linux die erste Karte automatisch erkennen lassen und die Parameter der zweiten mit *lilo* übergeben. Dabei müssen Sie allerdings sicherstellen, daß der Treiber die zweite Karte nicht aus Versehen zuerst findet, da sonst die andere überhaupt nicht konfiguriert wird. Das können Sie dadurch erreichen, daß Sie *lilo* eine `reserve`-Option übergeben, die den Kernel ausdrücklich anweist, den von der zweiten Karte belegten I/O-Bereich nicht zu testen.

Wenn Sie zum Beispiel Ihre zweite Karte auf die Adresse `0x300` konfiguriert haben, übergeben Sie dem Kernel folgende Parameter:

```
reserve=0x300,32 ether=0,0x300,eth1
```

Die `reserve`-Option stellt sicher, daß kein Treiber auf den angegebenen I/O-Bereich von `0x300` bis `0x31F` zugreift, wenn er das Vorhandensein eines Geräts prüft. Diesen Parameter können Sie auch benutzen, um bei der Konfiguration Ihrer ersten Karte den Autoprobing-Mechanismus zu umgehen:

```
reserve=0x340,32 ether=0,0x340,eth0
```

Um das Autoprobing für Ihre Ethernet-Karte vollständig auszuschalten, können Sie eine Basisadresse von `-1` angeben:

```
ether=0,-1,eth0
```

Der PLIP-Treiber

PLIP bedeutet *Parallel Line IP* und ist eine kostengünstige Art, zwei Maschinen miteinander zu vernetzen. Es verwendet den Parallelport sowie ein spezielles Kabel und erreicht Übertragungsgeschwindigkeiten von 10 bis 20 Kilobit/s.

PLIP wurde ursprünglich von der Frima Crynwr Inc. entwickelt. Das Design ist ziemlich genial (oder, wenn Sie es bevorzugen, hackermäßig): Lange Zeit waren die parallelen Ports des PC nur unidirektionale Druckerschnittstellen, das heißt über die 8 Datenleitungen konnten Daten an das Peripherie-Gerät ausgegeben werden, aber nicht umgekehrt. PLIP umgeht das Problem, indem es die fünf Statusleitungen der Schnittstelle für die Eingabe

verwendet. Dadurch ist es allerdings gezwungen, alle Daten in Nibbles (halben Bytes) zu übertragen. Diese Arbeitsweise wird PLIP-Modus 0 genannt. Heutzutage werden diese unidirektionalen Schnittstellen allerdings immer seltener. Aus diesem Grunde gibt es eine zweite PLIP-Variante names Modus 1, die die vollen 8 Bit des Ports nutzt.

Zur Zeit unterstützt Linux nur Modus 0. Während dies bei älteren Versionen nicht der Fall gewesen sein soll, ist der Treiber heute mit der ursprünglichen Implementation von Crynwr und dem Treiber im NCSA *telnet* kompatibel.(8) Um zwei Maschinen mit PLIP zu verbinden, benötigen Sie ein spezielles Kabel, das unter anderem unter der Bezeichnung »Null-Printer« und »Turbo Laplink« verkauft wird. Sie können es allerdings auch mit relativ geringem Aufwand selbst zusammenlöten, die Steckerbelegung finden Sie in Anhang A.

Der PLIP-Treiber für Linux ist das geistige Kind beinahe zahlloser Personen und wird zur Zeit von Niibe Yutaka gepflegt. In den Kernel eingebunden, reserviert er eine Netzwerkschnittstelle für jeden der möglichen Druckerports, wobei *plip0* dem Port *lp0* (LPT1) entspricht, *plip1* dem Port *lp1* etc. Die Abbildung von Schnittstellen auf Ports stellt sich zur Zeit so dar:

Interface	I/O-Adresse	IRQ
plip0	0x3BC	7
plip1	0x378	7
plip2	0x278	5

Wenn Sie Ihre parallele Karte anders konfiguriert haben, müssen Sie diese Werte in *drivers/net/Space.c* ändern und den Kernel neu übersetzen.

Diese Zuordnungen bedeuten allerdings nicht, daß Sie den parallelen Port nicht wie üblich benutzen können. Er wird erst dann vom PLIP-Treiber reserviert, wenn Sie die entsprechende Netzwerkschnittstelle aktivieren.

Die SLIP- und PPP-Treiber

SLIP (*Serial Line IP*) und PPP (*Point-to-Point Protocol*) sind weitverbreitete Protokolle, mit denen IP-Pakete über serielle Verbindungen übertragen werden können. Eine ganze Reihe von Firmen, Universitäten und Vereinen bieten Internet-Zugang über SLIP- und PPP-Verbindungen und machen so IP-basierte Dienste auch für Privatpersonen erschwinglich.

Um SLIP oder PPP einzusetzen, sind keinerlei Manipulationen am Kernel vonnöten; Sie können dafür einen ganz gewöhnlichen seriellen Port benutzen. Da die Konfiguration der seriellen Schnittstelle aber nicht unmittelbar mit TCP/IP zu tun hat, wird sie in einem gesonderten Kapitel abgehandelt. Weitergehende Informationen schlagen Sie deshalb bitte in Kapitel 4, *Konfiguration der seriellen Hardware* nach.

Fußnoten

(1)

Die IRQs 2 und 9 bezeichnen denselben Kanal. Der PC besitzt zwei kaskadierte Interrupt-Bausteine mit jeweils 8 Kanälen, wobei der zweite Baustein mit IRQ2 des ersten verbunden ist.

(2)

Neuere Kernel-Versionen bieten auch Unterstützung für einen Paketfilter, der z.B. den Zugriff bestimmter Hosts auf diverse Netzwerkdienst einschränken kann.

(3)

Zweck der Übung ist es, eine Fragmentierung der Pakete auf Verbindungen zu vermeiden, die keine so großen Pakete transportieren können.

(4)

Dieser Treiber läßt sich auch als ladbares Kernel-Modul einsetzen.

(5)

Aus diesem Grunde wird das »dünne« Ethernet auch oft 10base2 genannt, im Gegensatz zum Thicknet (10base5), das eine Maximallänge von 500 Metern hat, wie auch im Gegensatz zu Twisted Pair (10baseT).

(6)

Paul können Sie unter der Adresse gpg109@rsphysse.anu.edu.au erreichen.

(7)

Sie ist zusammen mit anderen Patches für Laptops auf tsx-11.mit.edu im Verzeichnis packages/laptops erhältlich.

(8)

NCSA telnet ist ein beliebtes Programm für DOS-PCs, das TCP/IP über Ethernet und PLIP bietet und telnet und ftp unterstützt.

Kapitel 4
Konfiguration der seriellen Hardware

Gerüchten zufolge soll es dort draußen in Netzland Leute geben, die nur einen PC besitzen und auch nicht das nötige Kleingeld für einen 2Megabit-Anschluß ans Internet haben. Um trotzdem ihre tägliche Dosis an Mail und News zu bekommen, benutzen sie SLIP-Links, UUCP-Netze und Mailboxen, die auf dem normalen Telefonnetz aufbauen.

Dieses Kapitel soll all den Leuten helfen, die ihre Netzverbindung mit Hilfe von Modems aufrechterhalten. Es gibt allerdings viele Details, die dieses Kapitel nicht abdecken kann, z. B. die Konfiguration eines Modems zwecks Einwählens in Ihr System. All diese Themen werden im »Serial HOWTO« von Greg Hankins(1) behandelt, das regelmäßig nach *comp.os.linux.answers* gepostet wird.

Software für Modem-Verbindungen

Unter Linux gibt es eine ganze Reihe von Kommunikations-Programmen. Viele von ihnen sind einfache *Terminal-Programme,* die dem Benutzer bzw. der Benutzerin erlauben, sich in ein anderes System einzuwählen, als säße er bzw. sie vor einem ganz gewöhnlichen Terminal. Das traditionelle Terminal-Programm unter UNIX ist *kermit*, es ist allerdings etwas spartanisch. Es gibt wesentlich komfortablere Programme, die Nummernverzeichnisse, Script-Sprachen zum Anwählen und Einloggen in andere Systeme usw. unterstützen. Eines von ihnen ist *minicom*, dessen Oberfläche und Bedienung dem sehr ähnlich ist, was ehemalige DOS-Benutzer gewohnt sein mögen. Außerdem existieren noch X-basierte Terminal-Programme wie *seyon*.

Daneben gibt es für Leute, die eine Mailbox betreiben wollen, eine Reihe von BBS-Paketen für Linux. Einige davon sind auf **sunsite.unc.edu** unter */pub/Linux/system /Network* zu finden.

Abgesehen von Terminal-Programmen gibt es auch Software, die Daten ohne Ihr Eingreifen auf Ihren oder von Ihrem Rechner überträgt. Das Vorteilhafte daran ist, daß es wesentlich weniger Zeit kostet, einige Dutzend Kilobytes automatisch herunterzuladen, als Sie vielleicht dafür brauchen, um Ihre Mail online zu lesen oder eine Mailbox nach interessanten Artikeln zu durchforsten. Auf der Gegenseite erfordert dies allerdings auch mehr Plattenplatz, da dabei auch jede Menge Information übertragen wird, die Sie vielleicht gar nicht interessiert.

Der Inbegriff dieser Art von Kommunikations-Software ist UUCP. Dies ist ein Programmpaket, das Dateien von einem Host auf einen anderen übertragen und Befehle auf einem entfernten Rechner ausführen kann usw. Es wird häufig benutzt, um Mail und News in kleineren Netzen zu übertragen. Ian Taylors UUCP-Paket, das auch unter Linux läuft, wird in einem der folgenden Kapitel beschrieben. Andere, nicht-interaktive Software wird beispielsweise im Fido-Netz eingesetzt. Portierungen von Fido-Netzapplikationen wie z. B. *ifmail* sind unter Linux auch verfügbar.

Einführung in serielle Geräte

Die von einem UNIX-Kernel für die Kommunikation mit seriellen Geräten bereitgestellten Schnittstellen werden im allgemeinen *tty*s genannt. Dies ist eine Abkürzung für *Teletype*, den wichtigsten Hersteller von Terminals in den Anfangstagen von UNIX. Dieser Begriff wird heutzutage allgemein für zeichenorientierte Ein- und Ausgabegeräte benutzt. In diesem Kapitel werden wir ihn ausschließlich benutzen, um uns auf die Geräteschnittstellen des Kernel zu beziehen.

Linux unterscheidet drei Arten von Ttys: (virtuelle) Konsolen, Pseudo-Terminals (ähnlich einer bidirektionalen Pipe; wird von Applikationen wie *telnet* und *xterm* benutzt) und serielle Geräte. Die letzteren werden auch zu den Ttys gerechnet, weil sie interaktives Arbeiten über serielle Verbindungen erlauben, sei es von einem festverdrahteten Terminal aus oder über eine Telefonverbindung.

Ttys haben eine Reihe von Konfigurations-Parametern, die über den Systemaufruf *ioctl(2)* besetzt werden können. Viele davon beziehen sich ausschließlich auf serielle Geräte, da diese einen wesentlich höheren Grad an Flexibilität verlangen, um unterschiedliche Verbindungstypen bedienen zu können.

Zu den prominentesten Parametern gehören die Verbindungs-Geschwindigkeit und die Parität, aber es gibt auch so exotische Dinge wie Flags für die Umsetzung von Groß- in Kleinbuchstaben, von Wagenrücklauf in Zeilenvorschub usw. Der Tty-Treiber unterstützt auch mehrere Verbindungs-Modi (*line disciplines*), die das Verhalten des Treibers radikal ändern. Beispielsweise wurde der SLIP-Treiber unter Linux mit Hilfe eines solchen Verbindungs-Modus implementiert.

Der Begriff der Verbindungs-Geschwindigkeit ist nicht ganz einfach zu definieren. Der korrekte Ausdruck lautet Bitrate, was sich auf die Übertragungsgeschwindigkeit der Leitung in Bits pro Sekunde (kurz: bps) bezieht. Manchmal werden Sie auch hören, daß Leute den Ausdruck *Baud-Rate* verwenden, was nicht ganz korrekt ist. Diese Begriffe bezeichnen nicht ganz dasselbe. Die Baud-Rate bezieht sich auf eine physikalische Eigenschaft eines seriellen Geräts, nämlich auf die Taktrate, mit der elektrische Impulse übermittelt werden. Die Bitrate aber bezeichnet eine Eigenschaft einer bestehenden seriellen Verbindung, nämlich die durchschnittliche Anzahl der Bits, die pro Sekunde übertragen werden. Es ist wichtig zu wissen, daß diese Werte normalerweise stark voneinander abweichen, da die meisten Geräte mehr als ein Bit pro Impuls kodieren.

Auf serielle Geräte zugreifen

Wie auf alle Geräte in einem UNIX-System greifen Sie auch auf serielle Geräte durch spezielle Dateien im Verzeichnis */dev* zu. Es gibt zwei Arten von Gerätedateien, die mit seriellen Treibern zu tun haben, und für jeden Port gibt es jeweils eine Gerätedatei von jedem Typ. Abhängig von der Datei, über die Sie darauf zugreifen, wird sich das Gerät unterschiedlich verhalten.

Die erste Variante wird zum Einwählen in das System benutzt und trägt die Hauptnummer (*major number*) 4. Die Dateien heißen *ttyS0*, *ttyS1* etc. Die zweite Gruppe wird zum Auswählen verwendet; die Dateien heißen hier *cua0, cua1* usw. und tragen die Major-Nummer 5.

Die Gerätenummern (*minor numbers*) sind für beide Typen von Dateien gleich. Wenn Sie an einem der Ports *COM1* bis *COM4* ein Modem angeschlossen haben, ist seine Minor-Nummer die Nummer des COM-Ports plus 63. Wenn Sie eine andere Art von Hardware verwenden, zum Beispiel eine Multiport-Karte, schlagen Sie bitte im Serial-HOWTO nach.

Nehmen wir an, Ihr Modem sei an *COM2* angeschlossen. Dann ist seine Minor-Nummer 65, und die Major-Nummer zum Auswählen ist 5. Sie sollten in */dev* eine Datei namens *cua1* finden, die diese Gerätenummern trägt. Wenn Sie die seriellen Ttys in */dev* auflisten, finden Sie die Major- und Minor-Nummern in den Spalten 5 und 6:

```
$ ls -l /dev/cua*
```

```
crw-rw-rw-   1 root     root       5,  64 Nov 30  1993 /dev/cua0
crw-rw-rw-   1 root     root       5,  65 Nov 30  1993 /dev/cua1
crw-rw-rw-   1 root     root       5,  66 Nov 30  1993 /dev/cua2
crw-rw-rw-   1 root     root       5,  67 Jul  9  1994 /dev/cua3
```

Wenn keine solche Gerätedatei existiert, müssen Sie sie einrichten, indem Sie Superuser werden und folgende Befehle eintippen:

```
# mknod -m 666 /dev/cua1 c 5 65
# chown root.root /dev/cua1
```

Manche Leute empfehlen, einen symbolischen Link namens */dev/modem* einzurichten, der auf Ihr Modem zeigt, damit ungeübte Benutzer sich nicht den etwas unintuitiven Namen *cua1* merken müssen. Sie können allerdings nicht in dem einen Programm *modem* und in einem anderen den echten Gerätenamen verwenden, weil diese Programme sogenannte Sperr- oder Lock-Dateien verwenden, um anzuzeigen, daß das Gerät gerade benutzt wird. Per Konvention vergibt die Lock-Datei z. B. für *cua1* den Namen *LCK..cua1*. Wenn Sie nun einen anderen Namen für dasselbe Geräte benutzen, wird Ihr Programm diese Lock-Datei ignorieren und das Gerät trotzdem öffnen. Das Ende vom Lied ist, daß beide Applikationen nicht funktionieren.

Serielle Hardware

Linux unterstützt eine breite Palette an seriellen Karten, die den Standard RS-232 unterstützen. RS-232 ist der derzeit verbreitetste Standard für serielle Geräte in der PC-Welt. Er verwendet eine Reihe von Leitungen zur Übertragung der einzelnen Bits und zur Synchronisation. Weitere Leitungen können dafür benutzt werden, um das Anliegen eines Trägersignals zu signalisieren (bei Modems verwendet), sowie für den Handshake.

Obwohl den Hardware-Handshake vom Standard nicht vorgeschrieben wird, ist er doch sehr nützlich. Er erlaubt jeder der beiden Stationen, der anderen mitzuteilen, ob weitere Daten empfangen werden können oder ob die Gegenstelle eine Pause einlegen sollte, bis der Empfänger die anliegenden Daten verarbeitet hat. Die hierfür verwendeten Steuerleitungen heißen »Clear to Send« (CTS) und »Ready to Send« (RTS), woraus sich auch der umgangssprachliche Name RTS/CTS für den Hardware-Handshake ableitet.

In PCs wird die RS-232-Schnittstelle üblicherweise durch UART-Chips angesteuert, die vom National Semiconductor 16450 oder neueren Varianten wie dem NSC 16550A(2) abstammen. Einige Fabrikate (insbesondere interne Modems, die mit dem Rockwell-Chipsatz ausgerüstet sind) verwenden auch völlig andere Chips, die aber so programmiert worden sind, daß sie sich (fast) wie ein 16550 verhalten.

Der wesentliche Unterschied zwischen 16450s und 16550s ist, daß die letzteren einen 16 Byte großen FIFO-Buffer besitzen, während erstere nur einen 1 Byte großen Puffer ihr eigen nennen. Dadurch sind 16450s für Geschwindigkeiten bis 9600 bps geeignet, während Sie für höhere Geschwindigkeiten einen 16550 benötigen. Neben diesen unterstützt Linux auch den 8250-Chip, den ursprünglichen UART-Baustein im PC-AT.

In der voreingestellten Konfiguration überprüft der Kernel die Standard-Ports *COM1* bis *COM4* und weist ihnen, wenn vorhanden, die Minor-Nummern 64 bis 76 zu, wie oben beschrieben.

Wenn Sie Ihre seriellen Ports korrekt konfigurieren wollen, sollten Sie Ted Tsos *setserial*-Programm mitsamt dem Script *rc.serial* installieren. Dieses Script sollte beim Booten des Systems ausgeführt werden. Es verwendet *setserial*, um die seriellen Geräte im Kernel zu konfigurieren. Ein typisches *rc.serial* sieht so aus:

```
# /etc/rc.serial -- Konfigurations-Script für serielle Schnittstellen.
#
# Abfangen von "wild interrupts".
/sbin/setserial -W /dev/cua*
```

```
# Konfiguration der Schnittstellen./sbin/setserial /dev/cua0 auto_irq skip_test
autoconfig
/sbin/setserial /dev/cua1 auto_irq skip_test autoconfig
/sbin/setserial /dev/cua2 auto_irq skip_test autoconfig
/sbin/setserial /dev/cua3 auto_irq skip_test autoconfig
# Serielle Konfiguration anzeigen.
/sbin/setserial -bg /dev/cua*
```

Wenn Ihre serielle Karte nicht erkannt wird oder der Befehl **`setserial -bg`** eine falsche Einstellung anzeigt, werden Sie die korrekte Einstellung durch die Angabe der richtigen Werte erzwingen müssen. Besonders häufig scheinen Besitzer interner Modems mit dem Rockwell-Chipsatz mit diesen Problemen zu kämpfen haben. Wenn beispielsweise *setserial* behauptet, sie hätten einen 16450-UART, obwohl der Baustein in Wirklichkeit zum NSC 16550 kompatibel ist, müssen Sie den Konfigurations-Befehl des betreffenden Ports so abändern:

```
/sbin/setserial /dev/cua1 auto_irq skip_test autoconfig uart 16550
```

Ähnliche Optionen existieren, um die Auswahl eines bestimmten *COM*-Ports, der I/O-Adresse und des IRQ zu erzwingen. Bitte lesen Sie die Details in der Manpage *setserial(8)* nach.

Wenn Ihr Modem den Hardware-Handshake unterstützt, sollten Sie sicherstellen, daß dieser auch aktiviert ist. Überraschenderweise versuchen die wenigsten Kommunikations-Programme dies einzustellen, weshalb Sie es unter Umständen per Hand tun müssen. Das geschieht am besten auch im *rc.serial*-Script, indem Sie *stty* so aufrufen:

```
$ stty crtscts < /dev/cua1
```

Um zu überprüfen, ob der Hardware-Handshake eingeschaltet ist, geben Sie diesen Befehl ein:

```
$ stty -a < /dev/cua1
```

Dieser Befehl gibt den aktuellen Zustand aller Flags für dieses Gerät aus; wenn ein Flag mit einem vorangestellten Minus (wie in **`-crtscts`**) ausgegeben wird, besagt das, daß es ausgeschaltet ist.

Fußnoten

(1)
: Greg kann unter der Adresse gregh@cc.gatech.edu erreicht werden.

(2)
: Es gab auch einen NSC 16550, aber seine FIFO hat niemals richtig funktioniert.

Kapitel 5
TCP/IP-Konfiguration

In diesem Kapitel werden wir uns mit den Schritten befassen, die nötig sind, um Ihr Linux-System für TCP/IP einzurichten. Angefangen mit der Zuordnung von IP-Adressen arbeiten wir uns langsam durch die Konfiguration der Netzwerk-Schnittstellen und stellen einige Werkzeuge vor, die recht nützlich sind, um eventuellen Problemen mit Ihrer Netzwerk-Installation nachzuspüren.

Den Großteil der Tätigkeiten, die dieses Kapitel behandelt, werden Sie nur einmal erledigen müssen. Danach müssen Sie die meisten Konfigurations-Dateien nur noch anfassen, wenn Sie eine neue Maschine zu Ihrem Netz hinzufügen oder wenn sie Ihr System völlig neu konfigurieren. Einige der Befehle, die zur Initialisierung von TCP/IP dienen, müssen allerdings jedes Mal ausgeführt werden, wenn Ihr System bootet. Dies geschieht normalerweise durch die */etc/rc*-Skripten.

Gewöhnlich ist der netzwerkspezifische Teil der Boot-Prozedur in einem Skript namens */etc/rc.net* oder */etc/rc.inet* enthalten. Manchmal werden Sie auch zwei Skripten namens *rc.inet1* und *rc.inet2* vorfinden, wobei ersteres den Kernel-Teil des Netzwerk-Kodes initialisiert und letzteres die wichtigsten Netzwerk-Dienste und -Applikationen startet. Im folgenden werde ich mich an diese Unterteilung halten.

Zunächst werde ich die Aufgaben von *rc.inet1* beschreiben und die verschiedenen Applikationen für spätere Kapitel aufheben. Dieses Kapitel soll Ihnen dabei helfen, eine Befehlsfolge zusammenzustellen, die TCP/IP auf Ihrem System korrekt initialisiert. Diese Befehle sollten Sie in *rc.inet1* eintragen (und dabei eventuell vorhandene Beispiele löschen).

Einrichten des proc-Dateisystems

Die meisten Konfigurations-Tools der Net-3-Familie kommunizieren mit dem Kernel über das *proc*-Dateisystem, auch kurz *procfs* genannt. Das *procfs* ist eine Kernel-Schnittstelle, die den Zugang zu verschiedenen Laufzeit-Informationen des Betriebssystems über einen Dateisystem-ähnlichen Mechanismus bereitstellt. Wenn das *procfs* gemountet ist, können Sie seine Dateien wie die jedes anderen Dateisystems auflisten oder ihren Inhalt auslesen. Typische Beispiele sind die Dateien *loadavg*, die die Systemauslastung enthält, und *meminfo*, die die aktuelle Belegung des System- und Swap-Speichers anzeigt.

Dem fügt der Netzwerk-Kode das Verzeichnis *net* hinzu. Es enthält eine Reihe von Dateien, die Dinge wie die ARP-Tabellen, den Zustand aller TCP-Verbindungen und die Routing-Tabellen enthalten. Die meisten

Administrations-Werkzeuge aus dem Netzwerkbereich erhalten ihre Informationen aus diesen Dateien.

Das *procfs* wird gewöhnlich während des Sytemstarts am Verzeichnis */proc* gemountet. Die einfachste Methode ist, folgende Zeile in */etc/fstab* einzufügen:

```
# Geraet   Verzeichnis   Typ        Flags
  none     /proc         proc       defaults
```

Anschließend müssen Sie nur noch den Befehl »**`mount /proc`**« von Ihrer */etc/rc*-Datei aus aufrufen.

Heutzutage ist *procfs* in den meisten vorkompilierten Kerneln bereits eingebunden, da es auch von anderen Dienstprogrammen wie *ps* benutzt wird. Wenn Ihr Kernel *procfs* wider Erwarten nicht enthalten sollte, sehen Sie eine Meldung wie »**`mount: fs type procfs not supported by kernel`**.« In diesem Falle müssen Sie einen neuen Kernel bauen und mit »**`y`**« antworten, wenn Sie nach Unterstützung für das *procfs* gefragt werden.

Installation der Programme

Wenn Sie eine der vorkonfigurierten Linux-Distributionen benutzen, enthält sie sehr wahrscheinlich die größeren Netzwerk-Applikationen und Dienstprogramme mitsamt einem Satz von Konfigurations-Beispielen. Die einzige Gelegenheit, bei der Sie diese Programme unter Umständen durch neuere ersetzen müssen, ergibt sich bei der Installation eines neuen Kernels. Da neue Kernel-Versionen manchmal auch Veränderungen der Netzwerk-Schnittstelle mit sich bringen, kann es passieren, daß die alten Programme seltsame Fehlermeldungen bringen, nur halb funktionieren oder im harmlosesten Fall einfach die neuesten Funktionen noch nicht unterstützen. Das bedeutet auf jeden Fall, daß Sie Programme wie *ifconfig*, *route* etc. neu kompilieren müssen, manchmal benötigen Sie aber auch eine ganz neue Version. Sie finden diese Programme in einem Archiv namens *net-tools-XXX.tar.gz,* wobei *XXX* die Versionsnummer des frühesten Kernels ist, ab dem sie gültig sind.

Wenn Sie bestimme Standard-Applikationen selbst kompilieren und installieren wollen, finden Sie die Quellen dafür auf den meisten Linux FTP-Servern. Es handelt sich bei ihnen oft um mehr oder weniger heftig modifizierte Programme aus BSD Net2 oder BSD-4.4. Andere Applikationen wie *Mosaic*, *xarchie* oder *gopher* müssen Sie sich oft anderweitig besorgen. Die meisten lassen sich ohne Probleme umstandslos aus der Mailbox heraus kompilieren.

Die offizielle FTP-Site für Net-3 ist **sunacm.swan.ac.uk**, die von **sunsite.unc.edu** im Verzeichnis *system/Network/sunacm* gespiegelt wird. **sunsite** wiederum wird von mehreren deutschen FTP-Servern gespiegelt, z. B. von **ftp.gwdg.de** unter */pub /linux/mirrors/sunsite*. Der ISDN-Treiber von Mathias Urlichs ist von **ftp.uni-stuttgart.de** im Verzeichnis */pub/systems/linux/isdn* erhältlich.

Noch ein Beispiel

Für den Rest des Buches möchte ich ein weiteres Beispiel vorstellen, das weniger komplex als die Groucho-Marx-Universität ist, aber wahrscheinlich eher mit den Aufgaben vergleichbar ist, denen Sie gegenüberstehen. Betrachten wir die Virtuelle Brauerei, eine kleine Firma, die, wie der Name schon sagt, virtuelles Bier braut. Um ihr Geschäft effizienter führen zu können, wollen die virtuellen Brauer ihre Computer vernetzen, die zufällig alle PCs sind, auf denen ein funkelnagelneues Linux 1.2 läuft.

Im selben Stockwerk gegenüber ist die Virtuelle Kellerei, die eng mit der Brauerei zusammenarbeitet. Sie hat ihr eigenes Ethernet. Natürlich wollen die beiden Firmen ihre Netze miteinander verbinden, sobald sie einsatzbereit

sind. Als ersten Schritt wollen sie ein Gateway einrichten, das Pakete zwischen den beiden Netzen überträgt. Später möchten sie auch eine UUCP-Verbindung zur Außenwelt aufbauen, durch die sie Mail und News transportieren wollen. Längerfristig wünschen sie sich auch eine SLIP-Verbindung ins Internet.

Setzen des Hostnamens

Die meisten, wenn nicht gar alle Netzwerk-Applikationen verlassen sich darauf, daß der Name Ihrer Maschine einen sinnvollen Wert hat. Er wird im Normalfall während der Boot-Prozedur mit dem Befehl *hostname* gesetzt. Um den Hostnamen auf *foo* zu setzen, rufen Sie so auf:

```
# hostname foo
```

Es ist üblich, hier den unqualifizierten Namen ohne jede Domain-Angabe zu verwenden. Die Rechner der Virtuellen Brauerei könnten beispielsweise **vlager.vbrew.com**, **vale.vbrew.com** usw. heißen. Das sind ihre offiziellen, voll qualifizierten Hostnamen. Die jeweils erste Komponente wird nun oft für den lokalen Hostnamen benutzt, wie z. B. **vlager**. Da dieser lokale Name aber häufig dann verwendet wird, wenn eine Applikation die IP-Adresse des Rechners, auf dem sie läuft, herausfinden will, müssen Sie sicherstellen, daß der Resolver diesen Namen kennt. Das bedeutet im allgemeinen, daß Sie diesen Namen in der Datei */etc/hosts* eintragen müssen (mehr dazu gleich).

Manche Leute schlagen vor, mit dem Kommando *domainname* dem Kernel die Domain-Komponente des Hostnamens mitzuteilen, so daß man anschließend die Ausgabe von *hostname* und *domainname* einfach nur kombinieren müßte, um wieder den vollständigen Namen zu erhalten. Das ist bestenfalls zur Hälfte richtig, da der Befehl *domainname* eigentlich nur die NIS-Domain Ihres Systems festlegt, die mit der DNS-Domain, der Ihr Rechner angehört, nicht viel zu tun haben muß. (NIS wird in Kapitel 10, *Das »Network Information System«* behandelt.)

Neuere Versionen des *hostname*-Befehls sind in der Lage, den voll qualifizierten Hostnamen aus dem lokalen Namen abzuleiten.

IP-Adressen zuweisen

Wenn Sie Ihren Rechner für den Standalone-Betrieb konfigurieren (zum Beispiel, um nur das Newssystem INN laufen zu lassen), können Sie diesen Abschnitt getrost überspringen. In diesem Falle brauchen Sie nur eine einzige IP-Adresse für das Loopback-Interface, die immer **127.0.0.1** lautet.

In echten Netzen wie einem Ethernet liegen die Dinge ein wenig komplizierter. Wenn Sie sich an ein existierendes Netz anschließen wollen, müssen Sie dessen Administratoren bitten, Ihnen eine IP-Adresse zuzuweisen. Wenn Sie aber Ihr eigenes Netz aufbauen, müssen Sie selber die Adressen verteilen, wie im folgenden näher beschrieben wird.

Alle Hosts innerhalb eines physikalischen Netzes sollten üblicherweise Adressen aus demselben logischen IP-Netz verwenden. Daher müssen Sie Ihrem Netz zuerst eine IP-Netzwerk-Adresse zuweisen. Wenn Ihre Installation aus mehreren physikalisch unabhängigen Teilnetzen besteht, benötigen Sie für jedes Teilnetz eine eigene Adresse. In den allermeisten Fällen werden Sie zu diesem Zweck den zur Verfügung stehenden Adreßbereich in mehrere Subnetze unterteilen.

Was für eine IP-Adresse Sie wählen, hängt stark davon ab, ob Sie für die nähere Zukunft den Sprung ins Internet planen oder nicht. Wenn ja, sollten Sie bereits jetzt eine offizielle Internet-Adresse beantragen, da Ihnen das später eine mühselige Neueinrichtung Ihres Netzes erspart. Der beste Weg ist, Ihren Service-Anbieter dabei um Hilfe zu bitten. Wenn Sie eine IP-Adresse nur für den Fall beantragen wollen, daß Sie Ihr Netz irgendwann einmal ans

Internet anschließen werden, sollten Sie sich einen IP-Adreßantrag von Ihrem Netzwerk-Provider besorgen.

Ist Ihr Netz nicht mit dem Internet verbunden, und wird es auch in Zukunft nicht sein, können Sie sich theoretisch eine völlig beliebige Adresse aussuchen. Sie müssen nur sicherstellen, daß keine Pakete aus Ihrem internen Netz ins Internet entkommen können. Um völlig sicherzugehen daß kein Schaden entsteht, selbst wenn dies einmal passieren sollte, sollten Sie eine der IP-Netze verwenden, die für die private Nutzung reserviert sind. Die Internet Assigned Numbers Authority (IANA -- so etwas wie die Zulassungsstelle für die Datenautobahn) hat mehrere Netzwerk-Nummern der Kategorien A, B und C reserviert, die Sie benutzen können, ohne sie registrieren zu müssen. Diese Adressen sind nur innerhalb Ihres privaten Netzes gültig und werden nicht zwischen echten Internet-Systemen geroutet. Die Adressen (definiert in RFC 1597) sind:

Kategorie	Netze
A	**10.0.0.0**
B	**172.16.0.0** bis **172.31.0.0**
C	**192.168.0.0** bis **172.31.255.0**

Der zweite und dritte Block enthält dabei jeweils 16 bzw. 256 Netze.

Allerdings ist es nicht nur dann sinnvoll, eine Adresse aus diesem Bereich zu wählen, wenn Ihr Netz völlig vom Internet abgeschnitten ist; ein solches Netz hilft Ihnen auch, den Zugriff von außen auf Maschinen in Ihrem Netz auf ein einzelnes Gateway zu beschränken. Innerhalb Ihres Netzes wäre das Gateway über seine interne Adresse von allen Maschinen aus erreichbar, während es der Außenwelt unter seiner offiziellen Adresse bekannt wäre. Fremde Systeme könnten aber im allgemeinen Rechner innerhalb Ihres Netzes nicht erreichen, da deren Adressen im Internet nicht geroutet werden.

Im folgenden werden wir annehmen, daß die Netzwerk-Administratorin der Brauerei eine IP-Adresse der Kategorie B verwendet, z. B. **172.16.0.0**. Natürlich würde ein Netz der Klasse C vollkommen für alle Systeme der Brauerei und der Kellerei ausreichen. Ich verwende nur der Einfachheit halber ein Klasse-B-Netz, da es die Subnettierung leichter verständlich macht.

Einrichten von Subnetzen

Um mehrere Ethernets oder Token-Ringe in einem IP-Netz zu betreiben, müssen Sie Ihren Adreßbereich unterteilen. Bitte beachten Sie, daß eine Subnettierung nur notwendig wird, wenn Sie mehr als ein *Broadcast*-Netz haben; Punkt-zu-Punkt-Verbindungen wie SLIP und PPP zählen hier nicht. Wenn Sie beispielsweise ein Ethernet und eine oder mehrere SLIP-Verbindungen in die weite Welt unterhalten, besteht kein Bedarf an mehreren Subnetzen. Warum das so ist, werden wir später in Kapitel 7 erläutern.

Um zwei Ethernets bedienen zu können, entscheidet sich die Netz-Administratorin, 8 Bits des Host-Teils der Adresse als zusätzliche Subnetz-Bits zu verwenden. Damit bleiben weitere 8 Bits für den Host-Teil, womit pro Subnetz 254 Adressen möglich sind. Der Brauerei teilt sie das Subnetz 1 zu und der Kellerei Subnetz 2. Die jeweiligen Netzwerk-Adressen sind dann **172.16.1.0** und **172.16.2.0**, und die Netzmaske ist **255.255.255.0**.

vlager, das als Gateway zwischen den zwei Netzen fungiert, erhält auf beiden jeweils die Hostnummer 1, was eine IP-Adresse von **172.16.1.1** bzw. **172.16.2.1** ergibt. Abbildung 5--1 zeigt die beiden Subnetze und das Gateway.

Abbildung 5-1. Virtuelle Brauerei und Virtuelle Kellerei -- die zwei Subnetze.

Die Dateien hosts und networks

Nachdem Sie Ihr Netz subnettiert haben, benötigen Sie für den Anfang eine einfache Art der Namensauflösung, die auf der Datei */etc/hosts* basiert. Wenn Sie nicht vorhaben, DNS oder NIS zu benutzen, müssen Sie sogar alle Maschinen in Ihrem Netz in dieser Datei eintragen.

Aber auch, wenn Sie im normalen Betrieb DNS oder NIS einsetzen, ist es bequem, einen Teil der Rechnernamen in */etc/hosts* zu haben. Oft möchte man nämlich auch während des Bootens, wenn noch keine Netzwerk-Schnittstellen aktiv sind, symbolische Namen verwenden, beispielsweise im Skript *rc.inet1*. Sollten Sie einmal gezwungen sein, alle IP-Adressen zu ändern, müssen Sie nur noch die modifizierte *hosts*-Datei auf allen Rechnern installieren und die Systeme neu starten, anstatt sämtliche *rc*-Dateien von Hand zu editieren. Normalerweise werden Sie alle lokalen Rechner in */etc/hosts* eintragen, zusammen mit eventuell vorhandenen Gateways und NIS-Servern.

Außerdem sollten Sie während der anfänglichen Testphase nur Informationen aus der *hosts*-Datei verwenden. In manchen Distributionen enthalten die Konfigurations-Dateien für NIS und DNS Beispiele, die für seltsame Effekte sorgen können, wenn sie verwendet werden. Um sicherzustellen, daß alle Applikationen ausschließlich */etc/hosts* verwenden, wenn Sie die Adresse eines Systems suchen, müssen Sie die Datei */etc/host.conf* editieren. Kommentieren Sie alle Zeilen aus, die mit dem Schlüsselwort `order` beginnen, indem Sie ein Doppelkreuz voranstellen, und fügen Sie folgende Zeile ein:

```
order hosts
```

Die Konfiguration der Resolver-Bibliothek wird ausführlich in Kapitel 6, *Resolver und NameServer* behandelt.

Die Datei *hosts* enthält einen Eintrag pro Zeile, bestehend aus der IP-Adresse, dem Hostnamen und einer optionalen Liste von Aliases. Die Felder sind durch Leerzeichen oder Tabulatoren voneinander getrennt, und das Adreßfeld muß in Spalte eins beginnen. Ein Doppelkreuz leitet einen Kommentar ein.

Namen können entweder voll qualifiziert oder relativ zur lokalen Domain sein. Für **vale** würden Sie beispielsweise sowohl den vollständigen Namen **vale.vbrew.com** eintragen als auch **vale** ohne Domain-Angabe, so daß das System sowohl unter seinem offiziellen als auch unter dem kürzeren lokalen Namen bekannt ist.

Das folgende Beispiel zeigt, wie die Datei *hosts* in der virtuellen Brauerei aussehen könnte. Neben den normalen Namen sehen Sie zwei spezielle Einträge, **vlager-if1** und **vlager-if2**, die die Adressen für die beiden Schnittstellen des Gateway-Systems festlegen.

```
#
# Hosts-Datei fuer die Virtuelle Brauerei und die Virtuelle Kellerei
#
# IP             FQDN                    Aliase
#
127.0.0.1        localhost
#
172.16.1.1       vlager.vbrew.com        vlager vlager-if1
172.16.1.2       vstout.vbrew.com        vstout
172.16.1.3       vale.vbrew.com          vale
#
172.16.2.1       vlager-if2
172.16.2.2       vbeaujolais.vbrew.com vbeaujolais
172.16.2.3       vbardolino.vbrew.com  vbardolino
```

```
172.16.2.4      vchianti.vbrew.com    vchianti
```

Genau wie für IP-Adressen möchte man manchmal auch symbolische Namen für Netzwerk-Nummern verwenden. Aus diesem Grunde gibt es parallel zu */etc/hosts* die Datei */etc/networks,* um Netzwerknamen auf Adressen abzubilden und umgekehrt. In der Virtuellen Brauerei würden wir etwa folgende *networks*-Datei installieren:

```
#
# Networks-Datei fuer die Virtuelle Brauerei und Virtuelle Kellerei
#
brew-net        172.16.1.0
wine-net        172.16.2.0
loopback        127.0.0.0
```

Das Netz **loopback** kann für die Konfiguration der Loopback-Schnittstelle benutzt werden, wie weiter unten gezeigt wird.

Schnittstellen-Konfiguration für IP

Nachdem Sie Ihre Hardware wie im vorigen Kapitel beschrieben eingerichtet haben, müssen Sie sie der Netzwerk-Ebene des Kernels bekanntmachen. Zur Konfiguration der Schnittstellen und Initialisierung der Routing-Tabelle sind zwei Befehle von besonderer Bedeutung, nämlich *ifconfig* (wobei »if« für *Interface* steht) und *route*. Sie werden gewöhnlich aus *rc.inet1* heraus aufgerufen.

ifconfig dient dazu, eine Schnittstelle für die Netzwerkschicht des Kernels sichtbar zu machen. Das beinhaltet die Zuweisung einer IP-Adresse und verschiedener anderer Paremeter sowie die Aktivierung der Schnittstelle. Die einfachste Art, es auzurufen, ist:

```
# ifconfig interface ip-adresse
```

Das weist `interface` die Adresse `ip-adresse` zu und aktiviert es. Alle anderen Parameter werden auf voreingestellte Werte gesetzt. Die Netzmaske zum Beispiel wird aus der Kategorie der Adresse abgeleitet; für ein Klasse-B-Netz wäre das **255.255.0.0**. Wir werden uns am Ende des Kapitels noch ausführlicher mit *ifconfig* befassen.

route erlaubt Ihnen, Routen in die Routing-Tabelle des Kernels einzutragen oder aus ihr zu entfernen. Es kann aufgerufen werden als

```
# route [add|del] target
```

Dabei bestimmt das zweite Argument, ob die Route eingetragen oder entfernt wird.

Die Loopback-Schnittstelle

Die allererste Schnittstelle, die aktiviert werden muß, ist das Loopback-Interface:

```
# ifconfig lo 127.0.0.1
```

Manchmal werden Sie auch sehen, daß anstelle der IP-Adresse der Name localhost verwendet wird. *ifconfig* wird diesen Namen in der Datei *hosts* nachschlagen, wo er als Hostname für **127.0.0.1** definiert sein sollte:

```
# Eintrag fuer localhost
```

```
127.0.0.1       localhost
```

Um die Konfiguration einer Schnittstelle anzuzeigen, rufen Sie *ifconfig* nur mit dem Namen der Schnittstelle auf:

```
# ifconfig lo
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU:2000  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:0 errors:0 dropped:0 overruns:0
```

Wie Sie sehen, ist der Loopback-Schnittstelle die Netzmaske **255.0.0.0** zugewiesen worden, weil **127.0.0.1** eine Klasse-A-Adresse ist. Die Broadcast-Adresse ist auf den voreingestellten Wert gesetzt worden, in diesem Falle also **127.255.255.255**.

Frühe Kernels der Serie 1.1 setzten die Broadcast-Adresse der Loopback-Schnittstelle übrigens nicht automatisch, da das niemand sehr sinnvoll fand. Später stellte sich aber heraus, daß das sehr wichtig ist, falls das Kommando *rwho* richtig funktionieren soll.(1) Sollten Sie noch einen älteren Kernel fahren, aber Wert auf *rwho* legen, müssen Sie die Broadcast-Adresse von Hand setzen, was weiter unten im Abschnitt *Alles über ifconfig* beschrieben wird.

Jetzt können Sie fast schon anfangen, mit Ihrem »Mini-Netz« zu spielen. Was noch fehlt, ist ein Eintrag in der Routing-Tabelle, der festlegt, daß diese Schnittstelle als Route für das Netz **127.0.0.0** dient. Das erreichen Sie, indem Sie eingeben:

```
# route add 127.0.0.0
```

Natürlich können sie auch hier wieder anstelle der IP-Adresse den Namen **localhost** verwenden.

Als nächstes sollten Sie sich mit dem Programm *ping* davon überzeugen, daß alles einwandfrei funktioniert. *ping* ist das netztechnische Äquivalent eines Sonars(2) und wird benutzt, um festzustellen, ob ein System überhaupt erreichbar ist und wie lange ein Paket zu diesem System und zurück benötigt. Diese Zeit wird auch oft als *round-trip time* bezeichnet.

```
# ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=32 time=1 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=32 time=0 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=32 time=0 ms
^C
--- localhost ping statistics ---
3 packets transmitted, 3 packets received, 0 packet loss
round-trip min/avg/max = 0/0/1 ms
```

Wenn Sie *ping* wie hier gezeigt aufrufen, wird es fortwährend Pakete aussenden, bis Sie es unterbrechen. Das \verb|^C| oben markiert die Stelle, an der wir Ctrl-C gedrückt haben.

Das obige Beispiel zeigt, daß Pakete an **127.0.0.1** korrekt ausgeliefert werden und nahezu augenblicklich eine Antwort an *ping* zurückgeschickt wird. Das beweist, daß Sie Ihre erste Netzwerkschnittstelle erfolgreich konfiguriert haben.

Wenn *ping* eine völlig andere Ausgabe produziert als hier dargestellt, haben Sie ein Problem. Prüfen Sie, ob irgendwelche Fehlermeldungen nahelegen, daß eine Datei nicht korrekt installiert worden ist. Prüfen Sie

außerdem, ob die verwendeten Programme *ifconfig* und *route* mit Ihrer Kernel-Version kompatibel sind und vor allem ob der Kernel mit Netzwerk-Unterstützung übersetzt wurde (letzteres erkennen Sie an der Existenz des Verzeichnisses */proc/net*). Wenn Sie eine Fehlermeldung bekommen, die sinngemäß »Network Unreachable« lautet, haben Sie sich wahrscheinlich beim *route*-Kommando vertippt.

Die bisher beschriebenen Schritte reichen aus, um Netzwerk-Applikationen auf einem alleinstehenden Rechner zu benutzen. Nachdem Sie die zwei Aufrufe von *ifconfig* und *route* in *rc.inet1* eingetragen und sichergestellt haben, daß beide *rc.inet*-Skripten beim Systemstart ausgeführt werden, können Sie Ihre Maschine neu booten und einige Applikationen ausprobieren. Zum Beispiel sollte »**`telnet localhost`**« eine *telnet*-Verbindung aufbauen und Ihnen den Login-Prompt Ihres Systems geben.

Das Loopback-Interface ist allerdings nicht nur als Beispiel in Netzwerkbüchern oder als Testumgebung während der Software-Entwicklung nützlich, sondern wird auch von einigen Applikationen während des normalen Betriebs benutzt.(3) Aus diesem Grunde müssen Sie es auf jeden Fall konfigurieren, unabhängig davon, ob Ihre Maschine an ein Netz angeschlossen ist oder nicht.

Ethernet-Schnittstellen

Die Konfiguration von Ethernet-Schnittstellen geht fast genauso vonstatten wie beim Loopback-Gerät; Sie brauchen nur ein paar Parameter mehr, um Subnetze verwenden zu können.

In der virtuellen Brauerei haben wir das IP-Netz, das ursprünglich ein Klasse-B-Netz war, in C-Subnetze unterteilt. Um das dem Interface mitzuteilen, sähe die *ifconfig*-Anrufung so aus:

```
# ifconfig eth0 vstout netmask 255.255.255.0
```

Dies weist der Schnittstelle *eth0* die IP-Adresse von **vstout** zu (**172.16.1.2**). Hätten wir die Netzmaske weggelassen, hätte *ifconfig* sie aus der Netzklasse der Adresse abgeleitet, was einen Wert von **255.255.0.0** ergeben hätte. Ein schneller Test ergibt jetzt:

```
# ifconfig eth0
eth0      Link encap:10Mps Ethernet HWaddr  00:00:C0:90:B3:42
          inet addr:172.16.1.2 Bcast:172.16.1.255 Mask:255.255.255.0
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:0 errors:0 dropped:0 overruns:0
          Interrupt:12 Base address:0x320
```

Wie Sie sehen, hat *ifconfig* die Broadcast-Adresse (im **`Bcast`**-Feld angezeigt) automatisch auf den üblichen Wert gesetzt, nämlich die Netzwerk-Nummer mit einem Host-Teil, bei dem alle Bits auf eins gesetzt sind. Außerdem wurde die maximale Übertragungseinheit (MTU -- *Maximum Transmission Unit*) auf das Ethernet-spezifische Maximum eingestellt.(4) Alle diese Werte können über spezielle Optionen auch mit anderen Werten besetzt werden; diese Optionen werden wir weiter unten beschreiben.

Fast wie im Falle des Loopback-Geräts müssen Sie jetzt eine Route eintragen, die dem Kernel mitteilt, welches Netz durch *eth0* erreicht werden kann. Für die virtuelle Brauerei würden Sie *route* so aufrufen:

```
# route add -net 172.16.1.0
```

Auf den ersten Blick sieht das ein wenig wie Schwarze Magie aus, da nicht ohne weiteres klar ist, wie *route* eigentlich wissen soll, daß wir hiermit die Ethernet-Schnittstelle meinen. Der Trick ist aber relativ simpel: Der Kernel prüft alle bisher konfigurierten Interfaces und vergleicht das Zielnetz (in unserem Falle **172.16.1.0**) mit der

Netznummer der Interface-Adresse, d. h. dem bitweisen UND der Interface-Adresse und der Netzmaske. Die einzige Schnittstelle, bei der diese beiden Werte übereinstimmen, ist *eth0*.

Aber was soll dann die Option *-net*? Sie wird nötig, da *route* sowohl Routen zu ganzen Netzen als auch zu einzelnen Maschinen einrichten kann, wie Sie oben bereits am Beispiel der **localhost**-Route gesehen haben. Wenn Sie *route* eine IP-Adresse in Dezimalnotation übergeben, versucht es zu erraten, obe es sich dabei um eine Host- oder Netzadresse handelt, indem es den Host-Teil anschaut. Ist der Host-Teil Null, nimmt es an, daß die Adresse ein Netz bezeichnet, andernfalls behandelt es sie als Host-Adresse. Aus diesem Grunde würde *route* in unserem Beispiel davon ausgehen, daß **172.16.1.0** eine Hostadresse ist, denn es kann ja nicht wissen, daß wir Subnetze verwenden. Darum müssen Sie ihm ausdrücklich mitteilen, daß sie ein Netz bezeichnet, indem Sie *-net* verwenden.

Natürlich können Sie es auch einfacher haben, indem Sie zum Beispiel die Netznamen benutzen, die wir oben in */etc/networks* definiert haben. Das hat nicht nur die bekannten Vorteile bei einer Umstrukturierung Ihres Netzes, sondern macht den Befehl auch wesentlich lesbarer. Sie können nun sogar die Option *-net* weglassen, da *route* jetzt weiß, daß **brew-net** ein Netz bezeichnet.

```
# route add brew-net
```

Nachdem Sie die grundlegenden Konfigurations-Schritte hinter sich gebracht haben, sollten Sie überprüfen, ob Ihr Ethernet tatsächlich fröhlich vor sich hinbrummt. Wählen Sie irgendeine Maschine auf Ihrem lokalen Ethernet und geben Sie folgenden Befehl ein:

```
# ping vlager
PING vlager: 64 byte packets
64 bytes from 172.16.1.1: icmp_seq=0. ttl=255 time=3.1 ms
64 bytes from 172.16.1.1: icmp_seq=1. ttl=255 time=2.4 ms
64 bytes from 172.16.1.1: icmp_seq=2. ttl=255 time=4.0 ms
64 bytes from 172.16.1.1: icmp_seq=3. ttl=255 time=2.2 ms
^C
----vstout.vbrew.com PING Statistics----
4 packets transmitted, 4 packets received, 0 packet loss
round-trip (ms)  min/avg/max = 2.2/2.9/4.0
```

Wenn sich die Ausgabe auf Ihrem Bildschirm deutlich von dem unterscheidet, was Sie hier sehen, ist offensichtlich etwas faul. Ungewöhnlich hoher Datenverlust(5) beispielsweise legt ein Hardware-Problem nahe, wie beispielsweise ungenügende oder fehlende Terminierung. Wenn *ping* keinerlei Pakete zurückempfängt, sollten Sie die Schnittstellen-Statistik mit *netstat* überprüfen. Die Paket-Statistiken, die *netstat* ausgibt, sagen Ihnen, ob überhaupt Pakete über das Interface übertragen wurden. Bei manchen Karten »erkennt« Linux manchmal auch einen falschen Interrupt. Eine von Linux falsch erkannte Karte macht sich im allgemeinen dadurch bemerkbar, daß sich das Interface ohne Fehlermeldungen konfigurieren läßt, aber keine Pakete auf das Ethernet geschickt werden.

Wenn Sie außerdem Zugang zum Zielrechner Ihres *ping*-Versuchs haben, sollten Sie auch auf diesem die Paket-Statistiken Maschine prüfen. Auf diese Art können Sie genau feststellen, wo die Daten verlorengehen. Zusätzlich sollten Sie auf beiden Maschinen mit *route* die Routing-Informationen überprüfen. Wenn Sie *route* ohne weitere Parameter aufrufen, gibt es die Routing-Tabellen des Kernels aus. Die Option *-n* sorgt dafür, daß es Adressen in Dezimalnotation anstelle der symbolischen Hostnamen darstellt:

```
# route -n
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
127.0.0.0       *               255.0.0.0       U     1936   0      112 lo
```

```
193.175.30.32   *               255.255.255.248 U     1436   0        10 eth0
```

Die genaue Bedeutung der einzelnen Felder wird weiter unten in Abschnitt *Testen mit netstat* erklärt. Die mit **`Flags`** betitelte Spalte enthält eine Liste von Flags für jede Zieladresse. **`U`** ist für aktive Schnittstellen immer gesetzt. Ein Flag **`H`** in dieser Spalte besagt, daß die Zieladresse einen einzelnen Host bezeichnet. In diesem Beispiel ist **`H`** bei keiner der beiden Routen gesetzt, da sich beide auf ein Netz beziehen. Sollte allerdings **`H`** für einen Eintrag gesetzt sein, der eigentlich eine Netzwerk-Route sein sollte, müssen Sie beim Eintragen der Route die Option -*net* mit angeben.

Um festzustellen, ob eine von Ihnen eingegebene Route überhaupt benutzt wird, prüfen Sie, ob sich das Feld **`Use`** zwischen zwei Aufrufen von *ping* erhöht.

Routing durch ein Gateway

Im vorigen Abschnitt habe ich nur beschrieben, wie Sie eine Maschine auf einem isolierten Ethernet einrichten. Der Regelfall ist allerdings, daß mehrere Netze durch Gateways miteinander verbunden sind. Diese Gateways können einfach zwei oder mehrere Ethernets miteinander verbinden, können aber auch Ihr Tor zur Außenwelt, dem Internet, darstellen. Um den Dienst eines Gateways zu nutzen, müssen Sie dem Routing-Mechanismus zusätzliche Informationen geben.

Zum Beispiel sind die Ethernets der Virtuellen Brauerei und der Virtuellen Kellerei durch solch ein Gateway miteinander verbunden, nämlich **vlager**. Angenommen, **vlager** sei bereits konfiguriert, dann müssen wir auf Maschinen wie **vstout** nur noch eine weitere Route eintragen, die angibt, daß alle Maschinen auf dem Netz der Kellerei über **vlager** erreichbar sind. Der entsprechende Aufruf von *route* sieht so aus:

```
# route add wine-net gw vlager
```

Dabei gibt *gw* an, daß das folgende Argument ein Gateway bezeichnet. Natürlich muß jedes System auf dem Kellerei-Netz einen analogen Routing-Eintrag für das Brauerei-Netz haben. Ansonsten könnten Sie nur Pakete von **vstout** an **vbardolino** schicken, aber jede Antwort von **vbardolino** würde im großen Biteimer landen.

Dieses Beispiel beschreibt nur ein Gateway, das Pakete zwischen zwei isolierten Ethernets befördert. Nehmen Sie nun an, daß **vlager** außerdem eine Verbindung ins Internet hat, beispielsweise durch einen SLIP-Link. Dann wäre es natürlich wünschenswert, daß Pakete für *beliebige* Zieladressen, die nicht auf dem Brauerei-Netz liegen, an **vlager** weitergereicht werden. Das können Sie erreichen, indem Sie **vlager** zum Default-Gateway für **vstout** machen:

```
# route default wine-net gw vlager
```

Die Netzwerk-Adresse **default** ist eine Abkürzung für **0.0.0.0**, die Default-Route. Diesen Namen müssen Sie nicht in */etc/netqworks* eintragen, er ist in *route* fest eingebaut.

Sollten Sie hohe Verlustraten beobachten, wenn Sie eine Maschine hinter einem Gateway mit *ping* ansprechen, könnte das ein Hinweis auf ein verstopftes Netz sein. In diesem Falle sind Paketverluste nicht so sehr die Folge technischer Probleme, sondern rühren von einer kurzzeitigen Überlastung der routenden Maschinen her, die dazu führt, daß ankommene Pakete verzögert oder gar weggeworfen werden.

Konfiguration eines Gateways

Es ist ziemlich einfach, eine Maschine dafür einzurichten, Pakete zwischen zwei Ethernets auszutauschen. Nehmen Sie an, wir befänden uns wieder auf **vlager**, das mit zwei Ethernet-Karten ausgestattet ist, die jeweils mit einem der beiden Netze verbunden sind. Alles was sie tun müssen ist, beide Schnittstellen getrennt zu konfigurieren und

ihnen eine Adresse auf dem jeweiligen Subnetz zuzuweisen, und das war's. Dabei ist es recht nützlich, zusätzlich zum offiziellen Hostnamen **vlager** zwei Namen für die beiden Schnittstellen in */etc/hosts* zu definieren:

```
172.16.1.1      vlager.vbrew.com vlager
172.16.1.1      vlager-if1
172.16.1.2      vlager-if2
```

Mit diesen Einträgen lautet die Befehlsreihenfolge für die Einrichtung von **vlager** so:

```
# ifconfig eth0 vlager-if1
# ifconfig eth1 vlager-if2
# route add brew-net
# route add wine-net
```

Die PLIP -- Schnittstelle

Wenn Sie PLIP verwenden, um zwei Maschinen miteinander zu vernetzen, liegen die Dinge etwas anders als bei einem Ethernet. PLIP stellt im Gegensatz zu einem Ethernet nur eine sogenannte Punkt-zu-Punkt-Verbindung her, d. h. es sind nur zwei Hosts involviert.

Als Beispiel betrachten wir den Laptop einer Angestellten der Virtuellen Brauerei, der mittels PLIP mit **vlager** verbunden werden kann. Der Laptop selbst heißt **vlite** und hat nur einen parallelen Port. Beim Booten wird dieser als *plip1* registriert. Um die Verbindung zu aktivieren, muß die Schnittstelle *plip1*mit den folgenden Befehlen konfiguriert werden:(6)

```
# ifconfig plip1 vlite pointopoint vlager
# route add vlager
# route add default gw vlager
```

Der erste Befehl richtet das Interface ein und teilt dem Kernel mit, daß es sich dabei um eine Punkt-zu-Punkt-Verbindung handelt, deren anderes Ende die IP-Adresse **172.16.1.1** (**vlager**) hat. Der zweite installiert eine Route für **vlager**, und der dritte richtet die Default-Route mit **vlager** als Gateway ein. Auf **vlager** sind entsprechende Kommandos nötig (die Default-Route darf natürlich nicht auf **vlite** zeigen):

```
# ifconfig plip1 vlager pointopoint vlite
# route add vlite
```

Das Besondere an diesem Fall ist, daß die Schnittstelle *plip1* auf **vlager** keine eigene IP-Adresse haben muß, sondern Sie ihr auch die Adresse **172.16.1.1** zuweisen können. Dies ist eine wichtige Eigenschaft von Punkt-zu-Punkt-Verbindungen.(7) Bis jetzt haben wir das Routing vom Laptop zu den Netzen der Virtuellen Brauerei eingerichtet. Jetzt fehlt noch die Möglichkeit, **vlite** von jedem Rechner der Virtuellen Brauerei aus zu erreichen. Eine besonders mühselige Methode ist, auf jedem System eine spezielle Route einzutragen, die **vlager** als Gateway für **vlite** angibt:

```
# route add vlite gw vlager
```

Eine viel bessere Methode, mit solchen temporären Routen zurechtzukommen, ist dynamisches Routing. Das können Sie beispielsweise mit *gated*, einem sogenannten Routing-Dämon verwirklichen, der die

Routing-Informationen von **vlager** an alle anderen Maschinen im Netz verteilt. Dazu muß *gated* allerdings auf jeder Maschine im Netz installiert und konfiguriert werden, womit wir auch nicht viel gewonnen hätten.

Am einfachsten fahren Sie in einem solchen Fall mit *Proxy-ARP*. Mit dieser Methode tut **vlager** gegenüber allen Maschinen auf dem Brauerei-Netz so, als *sei* es **vlite**, indem es alle ARP-Anfragen mit seiner eigenen Ethernet-Adresse beantwortet. Die Folge davon ist, daß alle Pakete für **vlite** auf **vlager** landen, das sie dann an den Laptop weiterreicht. Wir werden im Abschnitt *Die ARP-Tabelle* auf Proxy-ARP zurückkommen.

Die SLIP-Schnittstelle

Obwohl SLIP genau wie PLIP auch nur eine einfache Punkt-zu-Punkt-Verbindung darstellt, ist die Konfiguration eines SLIP-Links wesentlich komplizierter. Zum Verbindungsaufbau gehört meist noch das Einwählen ins fremde System und die Umschaltung der seriellen Leitung auf SLIP-Betrieb. Obwohl für jeden dieser Schritte separate Programme existieren, können Sie all dies auch mit einem einzelnen Programm namens *dip* erreichen. *dip* und die Konfiguration einer SLIP-Verbindung werden detalliert in Kapitel 7, *Serial Line IP* beschrieben.

Die Dummy-Schnittstelle

Die Dummy-Schnittstelle ist ein wenig exotisch, aber trotzdem unter gewissen Umständen gut zu gebrauchen. Ihre wichtigste Anwendung findet sich auf isolierten Maschinen und solchen, deren einzige IP-Verbindung zur Außenwelt eine Wählverbindung, zum Beispiel über SLIP oder PPP ist. Genaugenommen sind letztere die meiste Zeit auch isoliert (im Englischen verwendet man hier den Ausdruck *standalone*).

Das Dilemma mit solchen »alleinstehenden« Maschinen ist, daß ihre einzige aktive Schnittstelle das Loopback-Interface ist, das immer die Adresse **127.0.0.1** trägt. Manchmal müssen Sie aber auch Daten an die »offizielle« IP-Adresse Ihres Rechners schicken können. Betrachten Sie den Laptop **vlite**, der für die Dauer dieses Beispiels vom Netz getrennt wurde. Ein Applikation auf **vlite** möchte nun Daten an eine andere Applikation auf **vlite** senden. Dazu sucht sie die Adresse von **vlite** in */etc/hosts* heraus, erhält **172.16.1.65** und schickt das Paket an diese Adresse. Da aber zur Zeit das Loopback-Interface als einziges aktiv ist, kann der Kernel gar nicht wissen, daß sich diese Adresse auf den lokalen Host bezieht! Die Folge davon ist, daß er das Paket als unzustellbar wegwirft und der Applikation eine Fehlermeldung zurückliefert.

An dieser Stelle nun tritt das Dummy-Gerät auf den Plan. Es löst das Dilemma, indem es als alter ego des Loopback-Interface dient, sozusagen als Halter einer zweiten IP-Adresse. In unserem Fall würden sie ihm einfach die Adresse **172.16.1.56** geben und eine Route eintragen, die auf dieses Interface verweist. Jedes Paket an **172.16.1.56** wird daraufhin lokal ausgeliefert. Die vollständige Sequenz der Befehle ist:

```
# ifconfig dummy vlite
# route add vlite
```

Alles über ifconfig

ifconfig kennt neben den bisher vorgestellten noch eine ganze Reihe weiterer Optionen, die wir nun im einzelnen besprechen werden. *ifconfig* wird normalerweise so aufgerufen:

```
ifconfig interface [addresse [parameter]]
```

interface ist der Name der zu konfigurierenden Schnittstelle, und `adresse` ist die IP-Adresse, die ihr zugewiesen

werden soll. Sie kann entweder in Dezimalnotation angegeben werden oder als Name, den *ifconfig* in */etc/hosts* nachschlägt.

Wenn *iconfig* nur mit dem Interface-Namen aufgerufen wird, gibt es die Konfiguration der Schnittstelle aus. Wird es ganz ohne Parameter aufgerufen, zeigt es alle bisher konfigurierten Schnittstellen; die Option *-a* zwingt zusätzlich die Anzeige der inaktiven. Ein Aufruf für die Ethernet-Schnittstelle könnte beispielsweise so aussehen:

```
# ifconfig eth0
eth0      Link encap:10Mbps Ethernet  HWaddr 00:40:05:12:C6:54
          inet addr:172.16.1.2  Bcast:172.16.1.0 Mask:255.255.255.0
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:136 errors:0 dropped:0 overruns:0
          Interrupt:15 Base address:0x340
```

Die Felder **`MTU`** und **`Metric`** zeigen die aktuellen Werte für die MTU und die Metrik der Schnittstelle. Die Metrik wird von einigen Betriebssystemen verwendet, um die Kosten verschiedener Routen miteinander zu vergleichen. Linux benutzt diesen Wert bisher nicht, definiert ihn aber trotzdem aus Gründen der Kompatibilität. Die von *ifconfig* angezeigten Flags entsprechen mehr oder weniger den Namen der Befehlsparameter und werden gleich behandelt.

Die Zeilen RX und TX zeigen, wie viele Pakete empfangen (RX -- *receive*) bzw. gesendet wurden (TX -- *transmit*), wie viele Fehler dabei auftraten, wie viele Pakete (beispielsweise aufgrund von Speichermangel) weggeworfen wurden und wie viele wegen eines Überlaufs verlorengingen. Ein Überlauf des Empfängers (engl. *overrun*) tritt dann auf, wenn Pakete schneller hereinkommen als der Kernel den jeweils letzten Interrupt bedienen kann.

Die folgende Liste zeigt die Parameter, die *ifconfig* versteht; die Namen der zugehörigen Flags stehen in Klammern. Optionen, die einen bestimmten Dienst der Schnittstelle einschalten, können auch mit vorangestelltem Minus (-) benutzt werden, um ihn wieder auszuschalten.

up

Dies aktiviert ein Interface für die IP-Schicht des Kernels. Diese Option wird impliziert, wenn auf der Kommandozeile eine Adresse angegeben ist. Die Angabe der Option allein initialisiert das Interface mit Default-Werten für die IP-Adresse etc., was normalerweise wenig sinnvoll ist.

(Diese Option entspricht den Flags **`UP`** and **`RUNNING`**.)

down

Dies markiert eine Schittstelle als inaktiv, d. h. unzugänglich für die Netzwerkschicht. Gleichzeitig werden alle Routen durch dieses Interface entfernt.

netmask

`maske` Dies weist der Schnittstelle eine Subnetz-Maske zu. Sie kann entweder als eine 32-Bit-Hexadezimalzahl oder als punktierter Quadrupel angegeben werden.

pointopoint

`adresse` Diese Option wird für Punkt-zu-Punkt-Verbindungen benutzt, die nur zwei Hosts miteinander verbinden. Sie wird beispielsweise für die Konfiguration von SLIP- und PLIP-Schnittstellen benötigt und teilt dem Kernel die IP-Adresse des anderen Systems mit.

(Wenn eine Punkt-zu-Punkt-Adresse gesetzt worden ist, zeigt *ifconfig* das Flag **`POINTOPOINT`**.)

broadcast

`adresse` Die Broadcast-Adresse wird normalerweise aus der Netzwerknummer gebildet, indem alle Bits des Host-Teils auf eins gesetzt werden. Einige IP-Implementationen (zum Beispiel von BSD 4.2 abstammende Systeme) verwenden eine andere Broadcast-Adresse; die Option *broadcast* dient dazu, Ihre Konfiguration an solch eine seltsame Umgebung anzupassen.

(Wenn dem Interface eine Broadcast-Adresse zugeordnet worden ist, gibt *ifconfig* das Flag **`BROADCAST`** aus.)

metric

`wert` Mit dieser Option können Sie der Schnittstelle einen Metrikwert zuordnen. Dieser Wert wird beispielsweise vom *Routing Information Protocol* (RIP) berücksichtigt, wenn es Routing-Tabellen für Ihr Netz erstellt.(8) Die Default-Metrik, die *ifconfig* einem Interface zuweist, ist Null. Wenn Sie das Routing in Ihrem Netz nicht mit RIP regeln, benötigen Sie diese Option überhaupt nicht; aber selbst wenn Sie einen RIP-Dämon einsetzen, werden Sie nur äußerst selten Verwendung dafür haben.

mtu

`bytes` Dies setzt die *Maximum Transmission Unit* MTU, d. h. die maximale Zahl von Bytes, die das Interface in einer Transaktion behandeln kann. Für Ethernets liegt der Defaultwert bei 1500; für SLIP beträgt er 296.

arp

Diese Option kann nur für Broadcast-fähige Netz wie Ethernet oder Ham-Radio verwendet werden. Sie ermöglicht die Benutzung von ARP, dem *Address Resolution Protocol*, zur Zuordnung von IP-Adressen zu physikalischen Adressen. Für Broadcast-Netze wird sie per Voreinstellung eingeschaltet.

(Ist ARP ausgeschaltet, zeigt *ifconfig* das Flag **`NOARP`**.)

-arp

Schaltet ARP explizit aus.

promisc

Versetzt die Schnittstelle in den sogenannten »promiskuösen« Modus. Auf Broadcast-Netzen hat das zur Folge, daß die Schnittstelle alle Pakete unabhängig davon empfängt, ob sie für einen anderen Host bestimmt sind oder nicht. Dadurch kann man den Netzwerk-Verkehr mit Paketfiltern wie *tcpdump* analysieren, um Netzwerk-Problemen nachzuspüren, denen anders nur schwer beizukommen ist.

Dieses sogenannte *Packet Snooping* (d. h. Schnüffeln) hat aber auch seine problematische Seite. Mit ziemlich einfachen Mitteln können potentielle Cracker den Datenverkehr Ihres Netzes nach Paßwörtern filtern oder noch viel bösartigere Dinge tun. Dagegen können Sie sich kaum schützen, es sei denn, Sie entfernen alle DOS-Boxen aus Ihrem Netz und lassen niemanden in die Nähe Ihres Ethernet-Kabels. Beides ist nicht sehr realistisch. Eine andere Möglichkeit ist, ein sicheres Authentisierungs-Protokoll zu benutzen, wie es Kerberos oder die SRA Login-Suite bietet.(9) Diese Option entspricht dem Flag **`PROMISC`**.

-promisc

Schaltet den promiskuösen Modus ab.

Testen mit netstat

Als nächstes wenden wir uns einem nützlichen Werkzeug zu, mit dem Sie die Konfiguration und die Aktivität Ihres Netzes überprüfen können. Es heißt *netstat* und ist eher eine ganze Sammlung von Werkzeugen, die in einem Programm zusammengepackt worden sind. Wir werden im folgenden jede seiner Funktionen in einem separaten Abschnitt besprechen.

Anzeigen der Routing-Tabellen

Wenn Sie *netstat* mit dem Flag *-r* aufrufen, gibt es die Routing-Tabellen des Kernels aus, ähnlich, wie Sie es oben bereits bei *route* gesehen haben. Die Option *-n* sorgt zusätzlich dafür, daß die Adressen anstatt als symbolische Hostnamen direkt ausgegeben werden. Das ist besonders nützlich, wenn Sie verhindern wollen, daß der Resolver langwierige Nachfragen übers Netz startet, zum Beispiel beim DNS- oder NIS-Server.

Auf **vstout** sieht der Aufruf ungefähr so aus:

```
# netstat -nr
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use Iface
172.16.1.0      0.0.0.0         255.255.255.0   U     0      0   478 eth0
172.16.2.0      172.16.1.1      255.255.255.0   UG    0      0   250 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0    50 lo
```

Die erste Spalte zeigt jeweils das Ziel der Route an; Spalte vier enthält verschiedene Flags, die mehr über den Typ der Route sagen. Beispielsweise besagt das Flag `G` in der zweiten Zeile, daß es sich um eine Netzwerk-Route durch ein Gateway handelt. Das zugehörige Gateway ist in der zweiten Spalte angegeben. Geht die Route nicht durch ein Gateway, enthält die Spalte den Wert **0.0.0.0**.(10)

Die dritte Spalte gibt die »Allgemeinheit« der Route an. Bei Routen, deren Ziel ein Netz ist, entspricht das der Netzmaske. Host-Routen haben eine Genmask von **255.255.255.255**, und die Default-Route hat eine Maske von **0.0.0.0**. Die Routing-Tabelle ist so sortiert, daß spezielle Routen (lange Genmask) vor einer allgemeineren (kurze Genmask) stehen. Sucht der Kernel nun eine passende Route zu einer gegebenen Zieladresse, nimmt er das bitweise UND der Zieladresse und der Genmask und vergleicht das Resultat mit dem Ziel der Route.

Die vierte Spalte zeigt verschiedene Flags, die die Route näher charakterisieren.

G

Die Route geht durch ein Gateway.

U

Das zu verwendende Interface ist aktiv. Das ist bei \linux{} eigentlich immer der Fall.

H

Die Route zeigt auf einen einzelnen Host.

D

Der Eintrag wurde durch einen »ICMP Redirect« erzeugt. Diese Nachricht wird im allgemeinen von einem Gateway generiert, wenn es bemerkt, daß ein Host das Paket ohne Umweg direkt an die nächste Station der Route hätte schicken können, ohne dazu das Gateway zu benutzen.

M

Der Tabelleneintrag wurde durch einen ICMP Redirect modifiziert.

Die Spalte `Ref` sollte die Anzahl der Referenzen auf diese Route zeigen, ist aber zur Zeit bedeutungslos. Die beiden letzten Spalten zeigen, wie viele Päckchen bereits durch sie verschickt bzw. empfangen wurden und durch welches Interface sie führt.

Anzeige der Interface-Statistiken

Wenn Sie *netstat* mit dem Flag *-i* aufrufen, gibt es die Statistiken für die gerade aktiven Netzwerk-Schnittstellen aus. Geben Sie außerdem das Flag *-a* mit an, werden *alle* im Kernel vorhandenen Schnittstellen ausgegeben, nicht nur die konfigurierten. Auf **vstout** produziert *netstat* in etwa folgendes:

```
$ netstat -i
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR  TX-OK TX-ERR TX-DRP TX-OVR Flags
lo        0   0    3185      0      0      0   3185      0      0      0 BLRU
eth0   1500   0 972633     17     20    120 628711    217      0      0 BRU
```

Die Spalten **MTU** und **Met** geben die aktuelle MTU und Metrik des Interface an. Die mit **RX** bzw. **TX** überschriebenen Spalten geben an, wie viele Pakete fehlerfrei empfangen bzw. gesendet wurden (**RX-OK**/**TX-OK**), wie viele beschädigt waren (**RX-ERR**/**TX-ERR**), wie viele wegeworfen werden mußten (**RX-DRP**/**TX-DRP**) und wie viele aufgrund eines Overruns verlorengingen (**RX-OVR**/**TX-OVR**).

Die letzte Spalte zeigt wieder die Liste der Flags, die für die Schnittstelle gesetzt sind. Das sind einbuchstabige Versionen der langen Flagnamen, die *ifconfig* ausgibt:

```
--------------
B  BROADCAST
L  LOOPBACK
M  PROMISC
O  NOARP
P  POINTOPOINT
R  RUNNING
U  UP
--------------
```

Aktive Verbindungen und Sockets

netstat bietet eine Reihe von Optionen, mit denen Sie aktive und passive Sockets auflisten können. Die Argumente *-t*, *-u*, *-w* und *-x* zeigen aktive TCP, UDP, RAW und UNIX-Sockets. Wenn Sie zusätzlich *-a* angeben, sehen Sie außerdem die Sockets, die gerade auf eine Verbindung warten, weil Sie die Funktion *listen()* aufgerufen haben. Auf diese Weise erhalten Sie eine Liste aller Server, die derzeit auf Ihrem System laufen.

Ein Aufruf von *netstat -ta* ergibt auf **vlager**:

```
$ netstat -ta
Active Internet connections
Proto Recv-Q Send-Q Local Address    Foreign Address    (State)
tcp        0      0 *:domain         *:*                LISTEN
tcp        0      0 *:time           *:*                LISTEN
tcp        0      0 *:smtp           *:*                LISTEN
tcp        0      0 vlager:smtp      vstout:1040        ESTABLISHED
tcp        0      0 *:telnet         *:*                LISTEN
tcp        0      0 localhost:1046   vbardolino:telnet  ESTABLISHED
tcp        0      0 *:chargen        *:*                LISTEN
tcp        0      0 *:daytime        *:*                LISTEN
```

```
tcp        0      0 *:discard          *:*                LISTEN
tcp        0      0 *:echo             *:*                LISTEN
tcp        0      0 *:shell            *:*                LISTEN
tcp        0      0 *:login            *:*                LISTEN
```

Man sieht, daß die meisten Server einfach auf eine einkommende Verbindung warten, da sie sich im Zustand **`LISTEN`** befinden. Die vierte Zeile zeigt allerdings eine SMTP-Verbindung von **vstout**, und die sechste Zeile besagt, daß eine ausgehende *telnet*-Verbindung zu **vbardolino** besteht.(11)

Wenn Sie *netstat* nur mit der Option *-a* aufrufen, zeigt es eine kombinierte Liste aller Sockets aus allen Familien.

Die ARP-Tabelle

Bei einigen Netzproblemen kann es durchaus aufschlußreich sein, einen Blick auf die ARP-Tabelle des Kernels zu werfen oder sie sogar zu verändern. Wenn beispielsweise ein Host auf Ihrem Ethernet auf *ping* nicht reagiert, können Sie an der ARP-Tabelle erkennen, ob zumindest die Verbindung und die Hardware einwandfrei funktionieren oder nicht. Auf **vlager** sieht die Ausgabe beispielsweise so aus:

```
# arp -a
IP address      HW type                 HW address
172.16.1.3      10Mbps Ethernet         00:00:C0:5A:42:C1
172.16.1.2      10Mbps Ethernet         00:00:C0:90:B3:42
172.16.2.4      10Mbps Ethernet         00:00:C0:04:69:AA
```

Hier ist alles in bester Ordnung: In der ersten Spalte sehen Sie die IP-Adressen verschiedener Maschinen, die von **vlager** aus vor kurzem angesprochen wurden; in der letzten Spalte finden sich die Hardware-Adressen der Ethernet-Karten dieser Hosts.

Wenn in einem Eintrag in der letzten Spalte bei wiederholten Aufrufen statt der Ethernet-Adresse der Text **`(incomplete)`** erscheint, deutet das darauf hin, daß die Maschine entweder abgestürzt ist oder die Hardware defekt oder falsch konfiguriert ist.

Neben Ethernet wird ARP beispielsweise auch auf AX.25-Amateurfunknetzen und Token Ring eingesetzt, die aber natürlich jeweils eine andere Art der Adressierung benutzen. *arp* zeigt den Typ der Adresse jeweils in Spalte zwei an. Per Voreinstellung zeigt *arp* nur die Einträge für Ethernet-Adressen an. Um die Liste der AX.25-Adressen zu bekommen, müssen Sie es so aufrufen:

```
# arp -a -t ax25
```

Außer *-a* versteht *arp* auch noch die Optionen *-s* und *-d*. Wenn Sie *arp* mit der Option *-d* aufrufen, entfernt es alle Einträge für einen bestimmten Host, z. B. **vstout**:

```
# arp -d vstout
```

Damit zwingen Sie den Kernel, beim nächsten IP-Paket für **vstout** dessen Adresse neu zu erfragen. Das kann nützlich sein, wenn ein falsch konfigurierter Host falsche ARP-Informationen ausgegeben hat (vorher müssen Sie natürlich den fehlkonfigurierten Host »reparieren«).

-s dient dazu, die Hardware-Adresse eines Host manuell in die ARP-Tabelle einzutragen. Das ist beispielsweise sinnvoll, wenn ARP-Anfragen nach diesem Host nicht funktionieren, sei es, daß dessen ARP-Treiber eine Macke hat oder daß ein zweiter Host im Netz sich irrtümlich mit dessen IP-Adresse identifiziert. Die Festverdrahtung von

Hardware-Adressen im ARP-Cache ist auch eine (wenn auch drastische) Maßnahme, um unerlaubt an Ihr Netz angeschlossene Maschinen daran zu hindern, sich als jemand anders auszugeben.

Dazu wird *arp* so aufgerufen:

```
# arp -s vstout 00:00:C0:90:B3:42
```

Dieser Befehl trägt **vstout** mit der Hardware-Adresse **00:00:C0:90:B3:42** in die Tabelle ein. Im Gegensatz zu normalen Einträgen wird er nicht nach einer gewissen Zeit wieder entfernt, sondern bleibt permanent, bis Sie den Rechner rebooten.

Die Option *-s* kann aber auch für Proxy-ARP benutzt werden. Das ist eine spezielle Technik, bei der ein Host als Stellvertreter (engl. *proxy*) für eine andere Maschine fungiert, indem er vorgibt, daß beide IP-Adressen auf ihn zeigen. Im Abschnitt über PLIP zum Beispiel diente **vlager** als Gateway für den Laptop **vlite**. Mit Proxy-ARP fügen Sie dem ARP-Cache auf **vlager** nun noch einen Eintrag hinzu, der der IP-Adresse von **vlite** die Hardware-Adresse von **vlager** zuordnet. Will **vstout** nun ein Paket an **vlite** schicken, erkundigt er sich über ARP nach der Ethernet-Adresse von **vlite** und erhält -- ohne Böses zu ahnen -- die Adresse von **vlager** zurück. Darauf schickt er das Paket direkt an **vlager**, der es umgehend an **vlite** weiterleitet.

Der korrekte Aufruf dafür sieht so aus (wobei **00:40:05:12:C6:54** die Adresse von **vlager** ist):

```
# arp -s vstout 00:40:05:12:C6:54 pub
```

Diese Verrenkungen ersparen Ihnen nun die Arbeit, auf jedem Host des Brauerei-Ethernets einen expliziten Routing-Eintrag für **vlite** zu machen. Natürlich funktioniert das nur, wenn **vlite**s Adresse auf einem IP-Subnetz liegt, an das auch **vlager** angeschlossen ist. Beispielsweise könnte **vstout** einen Proxy-Eintrag für eine Adresse auf dem Brauerei-Subnetz (**172.16.1.0**) veröffentlichen, aber niemals für eine auf dem Kellerei-Netz (**172.16.2.0**).

Eine weitere Anwendungsmöglichkeit für Proxy-ARP kann angezeigt sein, wenn sich in Ihrem Netz eine DOS-Maschine mit einer etwas mangelhaften IP-Implementation befindet, die mit Subnetzen nicht zurechtkommt. In diesem Fall können Sie auf Ihrem Gateway Proxy-Einträge für alle Hosts hinter Ihrem Gateway einrichten, damit die DOS-Box sich nicht selbst um das Routing kümmern muß.

Fußnoten

(1)

Ohne Broadcast auf dem Loopback-Gerät zeigt rwho nur Benutzer auf anderen Maschinen im Netz an, ist aber völlig blind für seinen eigenen Host.

(2)

Erinnert sich noch jemand an Pink Floyds »Echoes«?

(3)

Zum Beispiel verwenden alle RPC-basierten Applikationen die Loopback-Schnittstelle, wenn sie sich beim portmapper registrieren.

(4)

Der MTU-Wert gibt die maximale Größe eines Pakets an, das über dieses Interface übermittelt werden kann.

(5)

Auf einem stark belasteten Ethernet oder mit einer sehr langsamen Ethernet-Karte ist es möglich, daß dann und wann ein Paket verlorengeht.

(6)

Die Schreibweise pointopoint ist wirklich kein Tippfehler:-)

(7)

Vorsichtshalber sollten Sie allerdings Ihren PLIP- oder SLIP-Link erst dann einrichten, wenn Sie bereits alle Routen für andere Netze eingetragen haben. Bei einigen älteren Kernels könnte eine falsche Reihenfolge dazu führen, daß Ihre Netzwerkroute anstatt aufs Ethernet plötzlich auf die PLIP-Schnittstelle zeigte.

(8)

RIP wählt aus mehreren zur Verfügung stehenden Routen zu einem Zielsystem die jeweils »kürzeste« aus. Die Länge eines Pfades setzt sich dabei aus den Metrikwerten der einzelnen Teilverbindungen zusammen. Per Voreinstellung hat jeder solche Hop die Länge 1, einzelne Schnittstellen haben eine Metrik von 0. Mit der Option metric können sie alle Routen durch ein bestimmtes Interface »teurer« machen.

(9)

SRA ist auf ftp.tamu.edu im Verzeichnis /pub/sec/TAMU erhältlich.

(10)

Bei manchen Versionen von netstat auch einfach nur ein Sternchen *.

(11)

Die Richtung einer Verbindung können Sie anhand der auftauchenden Port-Nummern erkennen. Das Ziel einer telnet-Verbindung zum Beispiel ist immer der Port 23, den netstat hier mit seinem symbolischen Namen ausgibt, der in /etc/services definiert ist. Der Port auf dem Ausgangsrechner dagegen ist so gut wie nie ein bekannter (well-known) Service-Port, weshalb netstat auch nur eine normale Zahl anzeigt.

Inhaltsverzeichnis  **Kapitel 4** **Kapitel 6**

Kapitel 6
Resolver und NameServer

In Kapitel 2, *Aspekte der Netzwerarbeit mit TCP/IP* habe ich bereits kurz beschrieben, daß in einer TCP/IP-Umgebung verschiedene Dienste zur Verfügung stehen, die Host-Namen auf IP-Adressen abbilden. Die einfachste Variante ist, eine Liste aller Host-Namen in der Datei */etc/hosts* abzulegen, die dann von den Applikationen einfach nach dem gewünschten Namen durchsucht wird. Das Format dieser Datei wurde bereits in Kapitel 5, *TCP/IP-Konfiguration* beschrieben.

/etc/hosts ist aber nur für kleine LANs sinnvoll, die von einer einzelnen Person gewartet werden und keinerlei IP-Anschluß zur Außenwelt haben. Ansonsten würden die Größe der Datei und der Pflegeaufwand schnell den Rahmen des Erträglichen sprengen. Auf die Dauer fahren Sie da wesentlich besser mit DNS, dem *Domain Name System*.(1) Die Einrichtung eines Name-Servers kann recht mühselig sein, aber wenn Sie das einmal hinter sich gebracht haben, sind Änderungen in der Netztopologie fast problemlos.

Im Internet ist DNS ein Muß, ohne es kommen Sie nicht weit. Aber auch in isolierten LANs ist es manchmal sehr nützlich, bringt aber hier auch keine nennenswerten Vorteile gegenüber NIS. Für ein kleines Ethernet mit vielleicht einem Dutzend Maschinen wird man oft auf beides verzichten und sich mit einer einfachen *hosts*-Datei begnügen.

Wenn Sie nicht gerade ein Netz administrieren, sondern einfach nur Ihre Linux-Box an ein existentes LAN anschließen wollen, müssen Sie Ihr System an die dortigen Gegebenheiten anpassen. Dazu steht Ihnen die Resolver-Bibliothek zur Verfügung, die transparent auf die verschiedenen Dienste zugreift. Der erste Teil diese Kapitels wird sich daher mit der Konfiguration dieser Bibliothek befassen.

Im zweiten Teil wenden wir uns der Einrichtung eines Name-Servers zu. Auf Linux, wie auf den meisten anderen UNIXoiden Systemen auch, wird die Arbeit des Name-Servers von *named* verrichtet, *name-dee* ausgesprochen. *named* ist Teil der BIND-Software, einer Sammlung von DNS-Werkzeugen und Bibliotheks-Funktionen. BIND steht für *Berkeley Internet Name Domain Server* und kommt aus der BSD-Welt.

In diesem Kapitel kann ich Ihnen nur wenig mehr als einen groben Überblick über den Betrieb eines Name-Servers geben. Wenn Sie BIND in einem komplexeren Umfeld als nur einem kleinen LAN und vielleicht einem kleinen Internet-Link betreiben wollen, sollten Sie sich ein gutes Buch über BIND besorgen, zum Beispiel Cricket Lius »DNS and BIND«. Daneben finden Sie in den Sourcen zu BIND ebenfalls sehr aktuelle Informationen. Außer den Manual-Seiten und den Release-Informationen enthalten sie den »BIND Operator's Guide«, kurz BOG. Lassen Sie sich von dem Namen nicht abschrecken, es ist wirklich eine sehr nützliche Referenz. Schließlich gibt es noch die Newsgruppe *comp.protocols.tcp-ip.domains*, die ausschließlich für Fragen und Diskussionen um DNS da ist.

Die Resolver-Bibliothek

Das Gegenstück zum Name-Server ist die Resolver-Bibliothek, eine Gruppe von Funktionen, die bei Linux zur Standard-C-Bibliothek gehören. Ihre wichtigsten Routinen sind *gethostbyname* und *gethostbyaddr*, die alle zu einem Namen gehörigen IP-Adressen zurückliefern, bzw. den kanonischen Host-Namen zu einer gegebenen Adresse. Über die Datei *host.conf* können Sie einstellen, ob sie die gewünschten Informationen in */etc/hosts* nachschlagen, das DNS befragen oder die *hosts*-Datenbank des NIS-Servers.(2)

Die Teile der Bibliothek, die sich mit DNS beschäftigen, stammen ursprünglich aus den BIND-Quellen, die auch den Name-Server enthalten. Die letzte aktuelle Version von BIND war lange Zeit 4.8, seit kurzem ist aber auch das neue BIND-4.9 öffentlich zugänglich.

Seit Version 4.6.8 enthält die Linux-C-Bibliothek den Resolver-Kode von BIND-4.9. Es ist wichtig zu wissen, daß BIND-4.9 einen wesentlichen Punkt des Resolvers geändert hat, die sogenannte *search list*, die weiter unten beschrieben wird. In allen anderen Punkten sollte sich das Verhalten der beiden Versionen nicht unterscheiden.

Die Datei host.conf

Die wichtigste Datei, die die Funktion Ihres Resolvers kontrolliert, ist *host.conf*. Sie liegt im Verzeichnis */etc* und legt unter anderem fest, welche Dienste der Resolver in welcher Reihenfolge befragen soll.

Optionen in *host.conf* müssen in getrennten Zeilen erscheinen, wobei die Argumente durch Leerzeichen oder TABs voneinander getrennt sein müssen. Ein Doppelkreuz leitet einen Kommentar ein, der sich bis zum Zeilenende erstreckt. Die folgenden Optionen sind verfügbar:

order

Dieser Befehl bestimmt die Reihenfolge, in der der Resolver die verschiedenen Dienste ausprobiert. Die Liste darf die Schlüsselworte `bind` für DNS, `hosts` für */etc/hosts* und `nis` für NIS-Abfragen enthalten.

multi

legt fest, ob Hosts in der Datei */etc/hosts* mehrere Adressen haben dürfen; diese Eigenschaft wird im Englischen oft als *multi-homed* bezeichnet. Gültige Argumente sind `on` und `off`. Dieses Kommando hat keinerlei Auswirkungen auf DNS- und NIS-Anfragen.

nospoof

Wie im vorherigen Kapitel beschrieben, kann DNS über die Domain **in-addr.arpa** IP-Adressen auf den zugehörigen kanonischen Host-Namen abbilden. Nun könnte ein böswilliger DNS-Administrator versucht sein, die tatsächlichen Namen seiner Hosts zu verschleiern, zum Beispiel, um Sicherheitsvorkehrungen zu umgehen. Um sich vor so einem versuchten Schwindel (= *spoof*) zu schützen, kann der Resolver zusätzlich überprüfen, ob die Adresse tatsächlich zu den gültigen Adressen des so erhaltenen Namens gehört. Tut sie das nicht, verwirft der Resolver den Namen und gibt einen Fehler an die Applikation zurück. Dieses Verhalten wird mit `nospoof on` eingestellt.

alert

Wenn Sie diese Option mit `alert on` anstellen, meldet der Resolver jeden Spoof-Versuch an den *syslog*-Dämon.

trim

gibt einen Domain-Namen an, den der Resolver wenn möglich vom Host-Namen abschneiden soll, bevor er die einzelnen Dienste befragt. Das kann für die *hosts*-Datei nützlich sein, wenn Sie dort nur die lokalen Host-Namen ohne Domain eintragen wollen. Übergibt eine Applikation dem Resolver nun einen

Host-Namen, der auf den lokalen Domain-Namen endet, wird er »getrimmt«, so daß der Resolver ihn in */etc/hosts* finden kann. In einer DNS-Umgebung macht das Trimmen von Domain-Namen allerdings keinen Sinn.

Sie können den `trim`-Befehl mehrmals angeben, wodurch Sie auch Host-Namen aus mehreren Domains in */etc/hosts* mischen können. Spätestens jetzt wird es allerdings unübersichtlich.

Beispiel 6--1 zeigt eine Beispieldatei *host.conf* für **vlager**.

Beispiel 6-1. <figure id=X-807-2-host.conf.file>Beispieldatei für host.conf.

```
# /etc/host.conf
# Wir benutzen named, aber kein NIS.
order   bind hosts
# Mehrere Adressen sind erlaubt.
multi   on
# Wehret den Schwindlern und Betruegern  n
ospoof  on
# Lokale Trimmdomain (hier nicht wirklich noetig)
trim    vbrew.com.
```

Konfiguration der Name-Server-Aufrufe -- resolv.conf

Wenn Ihr Resolver DNS benutzen soll, müssen Sie ihm auch mitteilen, welche Server er benutzen soll. Dafür gibt es eine separate Datei names *resolv.conf*.

Die wichtigste Option in *resolv.conf* ist `nameserver`, die die Adresse eines Name-Servers angibt. Wenn Sie die Option mehrmals angeben, werden die Server in der angegebenen Reihenfolge ausprobiert. Deshalb sollten Sie unbedingt den zuverlässigsten Server zuerst eintragen. Wenn Sie keinen Name-Server eintragen, nimmt der Resolver an, daß einer auf der lokalen Maschine läuft. Gegenwärtig werden bis zu drei `nameserver`-Einträge unterstützt.

Zwei weitere Befehle, `domain` und `search`, gaben Domain-Namen an, die der Resolver an einen Namen anhängt, wenn die zugehörige Adresse beim ersten Versuch nicht gefunden wurde. Wenn Sie beispielsweise *telnet* benutzen, wollen Sie im allgemeinen nicht immer den voll qualifizierten Domain-Namen eintippen, sondern lieber so etwas wie **gauss** angeben und den Resolver den Domain-Namen **mathematics.groucho.edu** drankleben lassen.

Genau dafür ist der Befehl `domain` da. Mit ihm können Sie eine Default-Domain angeben, die immer dann angehängt werden soll, wenn ein Name nicht aufgelöst werden konnte. Im obigen Beispiel würde der Resolver DNS zuerst nach **gauss**. befragen, was natürlich schiefgeht, da es keine Toplevel-Domain dieses Namens gibt. Mit **mathematics.groucho.edu** als Default-Domain würde er die Anfrage dann mit dem Namen **gauss.mathematics.groucho.edu** wiederholen, was diesmal von Erfolg gekrönt wäre.

Prima, werden Sie jetzt vielleicht denken, aber sobald ich die Domain verlasse, bin ich wieder bei den langen Namen und tippe mir die Finger wund. Warum kann ich nicht Namen wie **quark.physics** benutzen, wo wir doch schon an der Groucho-Marx-Universität sind?

Hier kommt die Suchliste ins Spiel. Eine Suchliste kann mit dem Befehl `search` angegeben werden und ist eine Verallgemeinerung der Default-Domain. Während Sie mit `domain` nur eine einzelne Domain angeben dürfen, akzeptiert `search` eine ganze Liste davon, deren Einträge alle der Reihe nach durchprobiert werden,

bis ein gültiger DNS-Eintrag gefunden wird. Die einzelnen Namen der Liste müssen durch Leerzeichen oder TABs voneinander getrennt werden.

Die Befehle `search` und `domain` schließen einander aus und dürfen höchstens einmal auftauchen. Wenn keiner der beiden Befehle angegeben ist, versucht der Resolver die Default-Domain aus dem lokalen Host-Namen zu raten. Hat der Host-Name keinen Domain-Teil, wird als Default-Domain die Root-Domain angenommen.

Wenn Sie in *resolv.conf* eine Suchliste definieren, sollten Sie sich genau überlegen, welche Domains Sie dort eintragen. Die Resolver-Bibliotheken vor BIND-4.9 pflegten aus dem Domain-Namen eine Liste zusammenzubauen, wenn vom Administrator keine angegeben wurde. Diese Default-Liste enthielt die Default-Domain plus alle Ober-Domains. Das konnte zu Problemen führen, da DNS-Anfragen manchmal bei Name-Servern landeten, für die sie nie bestimmt waren.

Nehmen Sie an, Sie sitzen in der Virtuellen Brauerei und wollen sich auf **foot.groucho.edu** einloggen. Durch einen unglücklichen Ausrutscher Ihrer Finger schreiben Sie aber **foo** statt **foot**. Der Name-Server der GMU kennt aber keine Maschine namens **foo**, was er Ihrem Resolver auch mitteilt. Mit der alten Suchliste würde der Resolver nun fortfahren und in den Domains **vbrew.com** und **com** fahnden. Vor allem der letzte Fall ist problematisch, da es durchaus eine Domain namens **groucho.edu.com** geben kann. Wenn diese Domain dann auch noch einen Host namens **foo** enthält, landet Ihr *login*-Programm bei einer ganz anderen Maschine, als von Ihnen beabsichtigt.(3)

Für einige Applikationen können diese fehlerhaften Zuordnungen ein Sicherheitsproblem darstellen. Aus diesem Grund sollten Sie die Suchliste auf die Unterdomains Ihrer Organisation oder etwas vergleichbares beschränken. Im Fachbereich Mathematik der Groucho-Marx-Universität würde die Liste beispielsweise nur **maths.groucho.edu** und **groucho.edu** enthalten.

Wenn Ihnen all das zu verwirrend vorkommt, werfen Sie einen Blick auf die *resolv.conf*-Datei der Virtuellen Brauerei:

```
# /etc/resolv.conf
# Unsere eigene Domain:
domain         vbrew.com
#
# Der zentrale Name-Server
nameserver      172.16.1.1
```

Wenn Sie in dieser Konfiguration die Adresse von **vale** suchen, wird der Resolver erst **vale** nachzuschlagen versuchen, und wenn das fehlschlägt, **vale.vbrew.com**.

Robustheit des Resolvers

Wenn Sie ein LAN innerhalb eines größeren Netzes betreiben, sollten Sie auf jeden Fall einen zentralen Name-Server benutzen (falls verfügbar). Der Vorteil dieser Methode ist, daß die zentralen Server im Laufe der Zeit einen reichhaltigen Cache entwickeln, da alle DNS-Queries über sie abgewickelt werden. Dieses Schema hat allerdings einen gravierenden Nachteil: als vor einiger Zeit ein Brand das Backbone-Kabel unserer Uni beschädigte, kam alle Arbeit auf dem LAN des Fachbereichs zum Erliegen, da der Resolver keinen der Uni-weiten Name-Server mehr erreichen konnte. Die X-Terminals waren tot, der Drucker war nicht mehr ansprechbar, etc.

Obwohl es nun nicht gerade häufig passiert, daß Uni-Backbones in Flammen aufgehen, ist es durchaus sinnvoll,

Vorsichtsmaßnahmen gegen solche Fälle zu treffen.

Eine Möglichkeit ist, einen eigenen Name-Server aufzusetzen, der die lokale Domain bedient und alle Anfragen für andere Host-Namen an den zentralen Server weiterleitet. Natürlich funktioniert das nur, wenn Sie eine eigene Domain administrieren.

Ebensogut können Sie aber auch Ihre lokalen Maschinen zusätzlich in */etc/hosts* eintragen und in der Datei */etc/host.conf* »`order bind hosts`« anfügen, damit der Resolver im Notfall auf diese statischen Tabellen zurückfallen kann.

Der Einsatz von named

Das Programm, das auf den meisten UNIX-Maschinen das DNS bedient, heißt *named* (ausgesprochen *name-dee*). Dieser Server wurde ursprünglich für BSD entwickelt und ist für Anfragen von Benutzerprozessen und eventuell anderen Name-Servern zuständig. Die derzeit auf Linux-Rechnern noch am häufigsten eingesetzte Version scheint BIND-4.8.3 zu sein. Die neue Version, BIND-4.9.3, ist derzeit noch im Beta-Test; es exisitieren jedoch bereits verschiedene Linux-Portierungen.(4)

Sie hat eine ganze Reihe neuer Features, wie beispielsweise die Möglichkeit, Zonentransfers auf bestimmte Maschinen oder Netze zu beschränken. Details hierzu finden Sie in der Dokumentation, die dem Quellkode beiliegt.

Dieser Abschnitt setzt voraus, daß Sie schon etwas über die Funktionsweise von DNS, dem Domain Name System, wissen. Wenn die folgende Diskussion für Sie wie böhmische Dörfer klingt, können Sie in Kapitel 2 einige zusätzliche Informationen über die Grundlagen von DNS nachlesen.

named wird für gewöhnlich beim Booten des Rechners gestartet und läuft, bis Sie die Maschine wieder herunterfahren. Er liest zunächst die Datei */etc/named.boot*, die seine Konfiguration beschreibt, und eine Reihe weiterer Dateien, die die Zuordnung von Domain-Namen zu Adressen und ähnliches festlegen. Letztere werden auch Zonendateien (*zone files*) genannt. Ihr Format und ihre Bedeutung werden im folgenden Abschnitt erklärt.

Um *named* zu starten, geben Sie am Prompt einfach folgendes ein:

```
# /usr/sbin/named
```

named legt nach dem Start seine Prozeß-ID in der Datei */var/run/named.pid* ab und liest anschließend die oben beschriebenen Dateien, und falls nötig, lädt er Zonendateien von anderen Servern. Anschließend wartet er auf Port 53 auf einkommende Anfragen.(5)

Die Datei named.boot

Die Datei *named.boot* ist im allgemeinen sehr klein und enthält wenig mehr als Verweise auf die Master-Dateien, in denen sich Zonendaten befinden. Kommentare in der Bootdatei beginnen mit einem Semikolon und reichen bis zum Zeilenende. Bevor wir aber das Format von *named.boot* im Detail angehen, sollten Sie einen Blick auf die Beispieldatei für **vlager** im folgenden werfen, die in Beispiel 6--2 dargestellt ist.(6) *Beispiel 6-2.*

```
;
   ; /etc/named.boot fuer vlager.vbrew.com
   ;
```

```
directory     /var/named
;
;             domain                    file
;-----------------------------------------------------------
cache         .                         named.ca
primary       vbrew.com                 named.hosts
primary       localhost                 named.local
primary       0.0.127.in-addr.arpa      named.local-rev
primary       72.191.in-addr.arpa       named.rev
```

Die im Beispiel gezeigten Befehle `cache` und `primary` laden DNS-Daten in *named*. Diese Informationen werden jeweils der Master-Datei entnommen, die im zweiten Argument angegeben wird. Sie enthalten Listen von DNS-Resource-Records, die wir uns später genauer ansehen werden.

Im diesem Beispiel wurde *named* als primärer Name-Server für vier Domains konfiguriert, wie die *primary*-Anweisungen am Ende der Datei zeigen. Die erste Zeile weist *named* beispielsweise an, als primärer Server für die vbrew.com zu dienen und die Zonendaten aus der Datei *named.hosts*zu lesen. Die `directory`-Zeile teilt ihm mit, daß sich alle Zonendateien im Verzeichnis */var/named* befinden.

Der `cache`-Eintrag hat eine besondere Aufgabe und sollte auf nahezu allen Maschinen vorhanden sein, die einen Name-Server einsetzen. Er weist *named* einerseits an, einen Cache aufzubauen, und andererseits, die Adressen der Name-Server für die Root-Domain aus der Datei *named.ca* zu laden. Diese Adressen heißen im Jargon *root name server hints*; wir werden im folgenden noch einmal auf sie zurückkommen.

Hier ist eine Liste der wichtigsten Optionen, die Sie in *named.boot* benutzen können:

directory

Dieser Befehl gibt das Verzeichnis an, in dem sich die Zonendateien befinden. Die Namen der Dateien können relativ zu diesem Verzeichnis angegeben werden. Sie können auch mehrere Verzeichnisse angeben, indem Sie den `directory`-Befehl mehrfach benutzen.

Dem Linux File System Standard zufolge sollten Zonendaten immer in */var /named* abgelegt werden.

primary

Dieser Befehl benötigt einen Domain-Namen und einen Dateinamen als Argumente und erklärt den Server als autoritativ für diese Domain. Als primärer Server lädt *named* die Zoneninformation aus der angegebenen Master-Datei.

Im allgemeinen wird *named.boot* mindestens zwei `primary`-Einträge enhalten, nämlich um dem Namen **localhost** die Adresse **127.0.0.1** zuzordnen und umgekehrt.

secondary

Dieser Befehl benötigt einen Domain-Namen, eine Adreßliste und einen Dateinamen als Argumente. Er erklärt den Server zum sekundären Server für die angegebene Domain.

Ein sekundärer Server ist ebenfalls autoritativ für die fragliche Domain, liest aber seine Informationen nicht aus Dateien. Statt dessen versucht er, sie vom primären oder einem anderen sekundären Server zu laden. Die Adreßliste muß deshalb die IP-Adresse mindestens eines autoritativen Servers enthalten.(7) Der lokale Server probiert diese Name-Server einen nach dem anderen, bis er die Zonendaten erfolgreich übertragen konnte. Die Daten werden anschließend in der Backup-Datei abgelegt, die im letzten Argument angegeben wurde. Wenn keiner der angegebenen Server erreichbar ist, lädt *named* die Zoneninformation statt dessen aus der Backup-Datei, sofern vorhanden.

named versucht dann, die Zonendaten in regelmäßigen Abständen aufzufrischen. Dies wird später im Zusammenhang mit dem Resource-Typ SOA erklärt.

cache

Dieses Kommando benötigt eine Domain und einen Dateinamen als Argumente. Die Datei enthält die Adressen der Root-Server. *named* ignoriert beim Lesen dieser Datei alle Records außer A und NS. Das Domain-Argument sollte der Name der Root-Domain sein, also ein einfacher Punkt (.).

Diese Daten sind für *named* äußerst wichtig: Wenn die `cache`-Anweisung nicht in der Boot-Datei auftaucht, wird *named* überhaupt keinen lokalen Cache bereits gefundener Adressen anlegen. Das kann unter Umständen zu einer schweren Performance-Bremse werden und die Netzbelastung unnötig erhöhen. Außerdem ist *named* so nicht in der Lage, irgendwelche Root-Server zu erreichen und kann somit nur solche Adressen auflösen, für die er autoritativ ist. Eine Ausnahme von dieser Regel sind sogenannte *forwarding servers*, die im nächsten Punkt behandelt werden.

forwarders

Dieser Anweisung folgt eine Liste von Adressen von Name-Servern, die *named* befragen darf, wenn er selbst nicht in der Lage war, eine Adresse aufzulösen. Diese Server werden der Reihe nach ausprobiert, bis einer von ihnen die Anfrage beantwortet.

slave

Dieser Befehle macht Ihren Name-Server zu einem Slave-Server. Das bedeutet, daß er niemals selbst rekursive Anfragen durchführt, sondern immer an die Server weiterleitet, die in der `forwarders`-Anweisung angegeben sind.

xfrnets

Dieser Befehl wurde in BIND-4.9.3 eingeführt und gibt an, welche Hosts die Zonendaten Ihres Servers vermittels eines Zonentransfers abrufen dürfen. Es soll verhindern, daß Leute ohne weiteres an eine vollständige Liste aller Hosts Ihrer Domain kommen.(8)

`xfrnets` erwartet eine Liste von IP-Netzadressen, die als dotted quads geschrieben werden müssen. Optional können Sie eine Netzmaske angeben, die von der Adresse durch ein & getrennt wird (z. B. **172.17.5.0&255.255.255.0**).

Neben diesen Optionen gibt es noch eine Reihe weiterer, die hier aber nicht beschrieben werden. Dazu gehören `sortlist` und `domain` plus einige, die in Version 4.9.3 neu eingeführt wurden. Daneben gibt es die Anweisungen `$INCLUDE` und `$ORIGIN`, die in den eigentlichen Zonendateien benutzt werden können. Da sie nur selten gebraucht werden, werde ich sie hier auch nicht beschreiben.

Die Zonendateien

Zu jeder Master-Datei wie *named.hosts*, die *named* bearbeitet, gehört ein Domain-Name, der im folgenden als Ursprung (engl. *origin*) bezeichnet wird. Das ist der Domain-Name, der beim jeweiligen `primary-` bzw. `cache`-Kommando angegeben wurde. Eine Name, der in einer Konfigurationsdatei angegeben wird, heißt absolut, wenn er auf einen Punkt endet, andernfalls ist er relativ zum Ursprung. Der Domain-Name »`@`« bezieht sich auf den Ursprung selbst.

Die Daten, die die Zonendateien enthalten, sind in sogenannte Resource-Records oder kurz RRs aufgeteilt. Sie stellen die kleinste durch das DNS erhältliche Informationseinheit dar. Jeder Resource-Record hat einen Typ. Die A-Records beispielsweise bilden einen Namen auf eine Adresse ab, und ein CNAME-Record definiert einen Alias für einen anderen Domain-Namen. Beispiel 6--4 zeigt die Master-Datei *named.hosts* für die Virtuelle Brauerei.

In den Master-Dateien haben alle Recource-Records ein gemeinsames Format:

[*domain*] [*ttl*] [*class*] *type rdata*

Die einzelnen Felder werden durch Leerzeichen oder Tabs voneinander getrennt. Manche Einträge können über mehrere Zeilen fortgesetzt werden, wenn vor dem ersten Zeilenende eine öffnende Klammer steht und auf das letzte Feld eine schließende Klammer.(9)

Alles zwischen einem Semikolon und dem Zeilenende wird ignoriert.

domain

Dies ist der Domain-Name, für den der Eintrag gilt. Wenn kein Name angegeben ist, bezieht sich der RR auf die Domain des vorigen RR.

ttl

Um andere Name-Server dazu zu veranlassen, nach einiger Zeit gecachte Informationen wegzuwerfen, ist für jeden RR eine bestimmte Lebensdauer (*time to live*, kurz ttl) festgelegt. Das Feld `ttl` legt die Zeit in Sekunden fest, die der Eintrag gültig ist, nachdem er vom Server abgerufen wurde. Der Wert ist eine Dezimalzahl mit höchstens 8 Ziffern.

Wenn kein ttl-Wert angegeben ist, wird der im SOA-Record angegebene Minimalwert (siehe unten) verwendet.

class

Dies ist eine Adreßklasse, wie IN für IP-Adressen oder HS für Objekte in der Hesiod-Klasse. Für TCP/IP-basierte Rechner müssen die Einträge den Typ IN haben.

type

Dies beschreibt den Typ des RR. Die häufigsten Typen sind A, SOA, PTR und NS. Im folgenden werden die verschiedenen Typen im einzelnen beschrieben.

rdata

Der Rest des Eintrages enthält die Daten, die zu dem RR gehören. Das Format dieses Feldes hängt vom Typ des RR ab.

Das folgende ist eine unvollständige Liste von RRs, die in Zonendateien benutzt werden können. Es gibt noch ein paar mehr, die ich hier aber nicht erklären will. Sie sind experimentell und werden so gut wie nicht benutzt.

SOA

Dieser Record leitet die Daten einer Zone ein (SOA bedeutet *Start Of Authority*). Jede Datei, die durch den Befehl `primary` eigebunden wird, muß mit einem solchen RR beginnen. Das Datenfeld enthält die folgenden Felder:

Ursprung

Dies ist der kanonische Host-Name des primären Name-Servers der Domain. Er wird üblicherweise als absoluter Name angegeben.

Kontakt

Dies ist die E-Mail-Adresse der Person, die für diese Zone verantworlich ist, wobei das Zeichen &guilsinglright;@&guilsinglleft; durch einen Punkt ersetzt wurde. Wenn beispielsweise janet die verantworliche Person in der Virtuellen Brauerei ist, enthält dieses Feld `janet.vbrew.com`.

Seriennummer

Dies ist die Versionsnummer der Zonendatei, geschrieben als einzelne Dezimalzahl. Jedesmal,

wenn Daten in der Zone verändert werden, sollte diese Zahl inkrementiert werden.

Die Seriennummer wird von den sekundären Name-Servern verwendet, um festzustellen, ob sich die Zonendaten verändert haben. Um auf dem laufenden zu bleiben, fordert der sekundäre Server den SOA-Record des primären Servers in bestimmten Zeitabständen an und vergleicht die Versionsnummer mit der des gecachten SOA-Records. Wenn sich die Zahl verändert hat, lädt er die gesamten Zonendaten vom primären Server neu.

Refresh

Dies gibt die Zeit in Sekunden an, die ein sekundärer Server warten soll, bevor er den SOA-Record des primären Servers wieder überprüft. Dies ist wieder eine Dezimalzahl mit bis zu 8 Stellen.

Normalerweise ändern sich die Namen in einer Domain nicht allzu häufig, weshalb diese Zahl ungefähr einem Tag im Falle größerer Domains entsprechen sollte, bei kleineren möglicherweise auch mehr.

Retry

Dieser Wert legt die Zeitabstände fest, in denen ein sekundärer Server versuchen soll, den primären zu erreichen, wenn eine Anfrage oder eine Zonenauffrischung fehlschlagen. Der Wert sollte nicht zu klein gewählt werden, da sonst eine vorübergehende Störung des Servers oder ein Netzproblem dazu führen können, daß der sekundäre Server unnötig Netzressourcen verbrät. Eine oder eine halbe Stunde dürften eine gute Wahl sein.

Expire

Dieser Wert gibt die Zeit in Sekunden an, nach denen ein sekundärer Server schließlich alle Zonendaten wegwerfen sollte, falls es ihm nicht gelingt, den primären Server zu erreichen. Sie sollten diesen Wert normalerweise auf mindestens eine Woche setzen (d. h. 604800 Sekunden), aber auch ein Monat kann durchaus sinnvoll sein.

Minimum

Dies ist der ttl-Wert, der als Voreinstellung für RRs gewählt wird, die nicht ausdrücklich einen solchen Wert angeben. Der ttl-Wert gibt die maximale Zeit an, für die andere Name-Server einen RR in Ihrem Cache behalten dürfen. Dieser Wert betrifft nur gewöhnliche DNS-Abfragen und hat nichts mit der Zeit zu tun, nach der ein sekundärer Server versuchen sollte, die Zoneninformation zu aktualisieren.

Wenn sich die Topologie Ihres Netzes nicht allzu häufig ändert, dürfte eine Woche oder mehr ein guter Wert sein. Wenn sich einzelne RRs häufiger ändern, können Sie diesen jeweils kleinere TTLs individuell zuordnen. Wenn sich der Aufbau Ihres Netzes allerdings häufiger ändert, können Sie diesen Wert auch auf einen Tag (86400 Sekunden) setzen.

A

Dieser Record-Typ ordnet einem Host-Namen eine Adresse zu. Das Datenfeld enthält die Adresse in dotted-quad-Schreibweise.

Für jede Adresse sollte es immer nur einen A-Record geben. Der in diesem RR benutzte Host-Name ist der offizielle oder *kanonische* Host-Name. Alle anderen Namen sind Aliase und müssen mittels eines CNAME-Records auf den kanonischen Host-Namen abgebildet werden.

Allerdings ist es durchaus möglich, daß ein Host-Name mehrere A-Records besitzt.

NS

NS-Records dienen dazu, primäre oder sekundäre Name-Server einer Zone anzugeben. Ein NS-Record

zeigt auf einen Name-Server der angegebenen Zone; das Datenfeld enthält den Host-Namen des Servers.

Sie werden NS-Records in zwei Situationen begegnen. Sie werden einmal benutzt, wenn Autorität an eine untergeordnete Zone delegiert wird, andererseits tauchen sie in der untergeordneten Zone selbst auf. Die Liste der Server, die jeweils in der delegierenden und delegierten Zone aufgelistet werden, sollten übereinstimmen.

Um den Host-Namen, auf den der NS-Record verweist, auflösen zu können, wird unter Umständen ein zusätzlicher A-Record benötigt, der sogenannte Glue-Record (*glue* = Leim), der die IP-Adresse des Servers angibt. Glue-Records werden in der Zonendatei der Eltern-Domain benötigt, wenn der bezeichnete Server in der delegierten Domain selbst liegt.

CNAME

Dieser Record-Typ ordnet einem Alias für einen Host dessen kanonischen Host-Namen zu. Der kanonische Host-Name ist derjenige, für den die Zonendatei einen A-Record enthält. Aliase verweisen über einen CNAME auf diesen Host-Namen und sollten keine weiteren RRs besitzen.

PTR

Dieser Record bildet einen Host-Namen auf einen anderen ab, ähnlich dem CNAME-Record. Er wird verwendet, um Namen in der Domain in-addr.arpa (die IP-Adressen repräsentieren) auf den zugehörigen kanonischen Host-Namen abzubilden. Das Datenfeld dieses RR enthält den kanonischen Host-Namen.

MX

Dieser RR legt einen sogenannten Mail Exchanger für die angegebene Domain fest. Mail Exchanger werden ausführlich in Abschnitt *Mail-Routing im Internet* in Kapitel 13, *Elektronische Post* beschrieben. Die Syntax eines MX-Records sieht so aus:

[*domain*] [*ttl*] [*class*] MX *preference host*

Das weist *host* als Mail Exchanger für *domain* aus. Jedem Exchanger ist eine ganzzahlige Präferenz zugeordnet. Ein Mail-Transportprogramm, das ein Nachricht an *domain* ausliefern möchte, probiert alle als Exchanger eingetragenen Hosts der Reihe nach durch, bis es sie erfolgreich übertragen kann. Dabei beginnt es bei dem MX mit der niedrigsten Präferenz und arbeitet die Liste in aufsteigender Reihenfolge ab.

HINFO

Dieser Record enthält Informationen über die Hard- und Software des Systems. Seine Syntax ist:

```
[domain] [ttl] [class] HINFO hardware software
```

Das *hardware*-Feld beschreibt die Hardware, auf der das System läuft, in einem bestimmten Format. Der RFC »Assigned Numbers« (gegenwärtig ist RFC 1340 die aktuelle Version) enthält eine Liste gültiger Namen, die Sie in diesem Feld eintragen können. Wenn das Feld Leerzeichen enthält, müssen sie in doppelte Anführungszeichen gesetzt werden. Das Feld *software* bezeichnet das Betriebssystem des Hosts. Auch hier sollten gültige Namen aus RFC 1340 gewählt werden.

Die Master-Dateien

In den Beispielen 6--3 bis 6--6 sehen Sie Beispieldateien für den Name-Server der Virtuellen Brauerei, der auf **vlager** läuft. Da wir hier nur ein relativ simples Netz (ein einfaches LAN) betrachten, ist das Beispiel auch nicht allzu kompliziert. Wenn Ihre Anforderungen komplexer sind, und es Ihnen partout nicht gelingt, *named* zum Laufen zu bringen, empfehle ich Ihnen *DNS and BIND* von Cricket Liu und Paul Albitz.

Die Datei *named.ca* in Beispiel 6--3 zeigt, wie die Rootserver Hints aussehen könnte. Eine typische Hint-Datei enthält normalerweise ein rundes Dutzend Name-Server. Um eine aktuelle Liste der Root-Server zu bekommen, können Sie das Programm *nslookup* verwenden, das im nächsten Abschnitt beschrieben wird.(10)

Beispiel 6-3. Die Datei named.ca

```
;
; /var/named/named.ca          Root Server Hints fuer die Virtuelle
;                              Brauerei. Wir sind nicht im Internet,
;                              also brauchen wir keine Root-Server. Um
;                              diese Records zu aktivieren, entfernen Sie
;                              die Semikolons.
;
; .                99999999   IN    NS  NS.NIC.DDN.MIL
; NS.NIC.DDN.MIL   99999999   IN    A   26.3.0.103
; .                99999999   IN    NS  NS.NASA.GOV
; NS.NASA.GOV      99999999   IN    A   128.102.16.10
```

Beispiel 6-4. Die Datei named.hosts

```
;
; /var/named/named.hosts       Lokale Hosts in der Brauerei.
;                              Ursprung is vbrew.com
;  @                IN  SOA   vlager.vbrew.com. janet.vbrew.com. (
                           16          ; serial
                           86400       ; refresh: once per day
                           3600        ; retry:   one hour
                           3600000     ; expire:  42 days
                           604800      ; minimum: 1 week
                           )
                 IN  NS    vlager.vbrew.com.
;
; Mail an user@vbrew.com wird auf vlager verteilt.
                 IN  MX    10 vlager
; Das Ethernet der Brauerei
vlager           IN  A     172.16.1.1
vlager-if1       IN  CNAME vlager
; vlager ist auch der News-Server
news             IN  CNAME vlager
vstout           IN  A     172.16.1.2
vale             IN  A     172.16.1.3
; Das Ethernet der Kellerei
vlager-if2       IN  A     172.16.2.1
vbardolino       IN  A     172.16.2.2
vchianti         IN  A     172.16.2.3
vbeaujolais      IN  A     172.16.2.4
```

Beispiel 6-5. Die Datei named.rev

```
;
; /var/named/named.rev         Reverse Mapping unserer IP-Addressen
;                              Ursprung ist 72.191.in-addr.arpa.
;
@             IN  SOA   vlager.vbrew.com. janet.vbrew.com. (
                          16         ; serial
                          86400      ; refresh: once per day
                          3600       ; retry:   one hour
                          3600000    ; expire:  42 days
                          604800     ; minimum: 1 week
                          )
              IN  NS    vlager.vbrew.com.
; Brauerei
1.1           IN  PTR   vlager.vbrew.com.
2.1           IN  PTR   vstout.vbrew.com.
3.1           IN  PTR   vale.vbrew.com.
; Kellerei
1.2           IN  PTR   vlager-if1.vbrew.com.
2.2           IN  PTR   vbardolino.vbrew.com.
3.2           IN  PTR   vchianti.vbrew.com.
4.2           IN  PTR   vbeaujolais.vbrew.com.
```

Die Beispiele 6--6 und 6--7 enthalten eigentlich nichts weiter als den Eintrag für den Namen **localhost**, der für die Adresse **127.0.0.1** steht. Neben den eigentlich wichtigen Records (dem A-Record für **localhost** und dem PTR-Record für **1.0.0.127.in-addr.arpa**) enthalten beide Dateien jeweils noch je einen SOA- und NS-Record. Das sieht nach ziemlichem Overkill aus und ist es wohl auch.(11) Es ist auch nicht unbedingt zwingend, **localhost** in Ihrem DNS zu definieren, Sie können auch genauso gut einen Eintrag in der Datei */etc/hosts* machen. Natürlich müssen Sie den Resolver in *host.conf*auch so konfigurieren, daß er außer dem DNS auch die *hosts*-Datei konsultiert.

Beispiel 6-6. Die Datei named.local

```
;
; /var/named/named.local       Der localhost-Eintrag
;                              Ursprung ist localhost.
;
@                 IN  SOA   vlager.vbrew.com. janet.vbrew.com. (
                           16         ; serial
                           86400      ; refresh: once per day
                           3600       ; retry:   one hour
                           3600000    ; expire:  42 days
                           604800     ; minimum: 1 week
                           )
                  IN  NS   vlager.vbrew.com.
; Und hier der eigentliche Eintrag:
localhost.        IN  A    127.0.0.1
```

Beispiel 6-7. Die Datei named.local-rev

```
;
; /var/named/named.local-rev   Reverse Mapping des Netzes 127.0.0
;                              Ursprung ist 0.0.127.in-addr.arpa.
;
@             IN  SOA   vlager.vbrew.com. janet.vbrew.com. (
                        1          ; serial
                        360000     ; refresh: 100 hrs
                        3600       ; retry:   one hour
                        3600000    ; expire:  42 days
                        360000     ; minimum: 100 hrs
                        )
              IN  NS    vlager.vbrew.com.
1             IN  PTR   localhost.
```

Test Ihrer DNS-Konfiguration

Es gibt mehrere sehr schöne Tools, um die Konfiguration Ihres Name-Servers zu testen. Zwei davon sind in der BIND-Distribution enthalten: *dif* und *nslookup*. Ich werde in diesem Abschnitt nur letzteres beschreiben, da es eine recht bequeme interaktive Benutzerschnittstelle besitzt. Sie können es aber auch von der Shell aus benutzen, indem Sie es einfach so aufrufen:

```
$ nslookup Host-Name
```

nslookup befragt dann den in *resolv.conf* angegebenen Name-Server nach `Host-Name`. (Wenn Sie in dieser Datei mehr als einen Server angegeben haben, wählt *nslookup* einen davon per Zufall aus).

Der interaktive Modus ist allerdings wesentlich spannender. Sie können in diesem Modus nicht nur einzelne Host-Namen auflösen, sondern nach beliebigen DNS-Records suchen und die gesamte Zoneninformation einer Domain auflisten.

Wenn Sie es ohne Argumente aufrufen, gibt *nslookup* den Namen des verwendeten Name-Servers aus und geht in den interaktiven Modus. Hinter dem Prompt > können Sie nun beliebige Domain-Namen angeben. Per Voreinstellung fragt *nslookup* nach A-Records, d. h. denjenigen, die die zum Domain-Namen gehörige IP-Adresse enthalten.

Sie können den Record-Typ, nach dem DNS-sucht, verändern, indem Sie

```
> set type=typ
```

eingeben. Dabei muß `typ` einer der oben beschriebenen Record-Typen sein oder der String ANY.

Eine Unterhaltung mit *nslookup* könnte beispielsweise so aussehen:

```
$ nslookup
Default Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60
> sunsite.unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60
Non-authoritative answer:
```

```
Name:    sunsite.unc.edu
Address:  152.2.22.81
```

Wenn Sie einen Namen angeben, zu dem es keine A-Records gibt, aber andere Record-Typen gefunden wurden, gibt *nslookup* die Fehlermeldung No type A records found. Mit dem Kommando *set type* können Sie es aber dazu bringen, auch andere Typen als A anzuzeigen. Um beispielsweise den SOA-Record für unc.edu zu bekommen, würden Sie etwa folgendes eingeben:

```
> unc.edu
*** No address (A) records available for unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60
> set type=SOA
> unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60
Non-authoritative answer:
unc.edu
        origin = ns.unc.edu
        mail addr = shava.ns.unc.edu
        serial = 930408
        refresh = 28800 (8 hours)
        retry   = 3600 (1 hour)
        expire  = 1209600 (14 days)
        minimum ttl = 86400 (1 day)
Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU       internet address = 128.109.157.30
```

Ganz ähnlich geht es, wenn Sie nach MX-Records usw. fragen wollen.

```
> set type=MX
> unc.edu
Non-authoritative answer:
unc.edu preference = 10, mail exchanger = lambada.oit.unc.edu
lambada.oit.unc.edu     internet address = 152.2.22.80
Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU       internet address = 128.109.157.30
```

Wenn Sie den Suchtyp auf ANY setzen, gibt *nslookup* alle Resource-Records aus, die zu dem angegebenen Namen gehören.

Außer für das Debugging Ihrer Konfiguration können Sie *nslookup* aber auch noch verwenden, um die aktuelle Liste der Root-Name-Server zu erhalten. Sie können das tun, indem Sie nach allen NS-Records fragen, die für die Root-Domain definiert sind:

```
> set type=NS
> .
Name Server:  fb0430.mathematik.th-darmstadt.de
```

```
Address:  130.83.2.30
Non-authoritative answer:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL
Authoritative answers can be found from:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL
NS.INTERNIC.NET internet address = 198.41.0.4
AOS.ARL.ARMY.MIL        internet address = 128.63.4.82
AOS.ARL.ARMY.MIL        internet address = 192.5.25.82
AOS.ARL.ARMY.MIL        internet address = 26.3.0.29
C.NYSER.NET     internet address = 192.33.4.12
TERP.UMD.EDU    internet address = 128.8.10.90
NS.NASA.GOV     internet address = 128.102.16.10
NS.NASA.GOV     internet address = 192.52.195.10
NS.NASA.GOV     internet address = 45.13.10.121
NIC.NORDU.NET   internet address = 192.36.148.17
NS.NIC.DDN.MIL  internet address = 192.112.36.4
```

Um eine Übersicht über alle Befehle zu bekommen, die *nslookup* bietet, geben Sie am Prompt den Befehl *help* ein. Um das Programm zu verlassen, geben Sie einfach Ctrl-D ein.

Andere Nützliche Dinge

Schließlich gibt es noch einige andere nützliche Tools, die Ihnen das Leben als BIND-Administrator leichter machen sollen. Ich beschreibe zwei von ihnen ganz kurz. Wenn Sie wissen wollen, wie diese Programme bedient werden, lesen Sie bitte die Dokumentation, die dem Source beiliegen sollte.

hostcvt ist ein Programm, das Ihnen bei Ihrer anfänglichen BIND-Konfiguration helfen soll, indem es Ihre *hosts*-Datei in Master-Dateien für *named* umsetzt. Es erzeugt sowohl A- als auch PTR-Records und achtet auf Aliase und ähnliches. Natürlich kann es nicht Ihren gesamten Job übernehmen, da Sie vielleicht noch einige der Voreinstellungen im SOA-Record anpassen oder MX-Records eintragen wollen und was dergleichen mehr ist. Trotzdem sparen Sie mit diesem Werkzeug vielleicht ein paar Aspirin. *hostcvt* ist Teil der BIND-Distribution, ist aber auch als einzelnes Paket auf einigen Linux-FTP-Servern zu finden.

Nachdem Sie Ihren Name-Server aufgesetzt haben, möchten Sie vielleicht Ihre Konfiguration auf Herz und Nieren überprüfen. Das ideale (und meines Wissens auch einzige) Tool ist *dnswalk*, ein *perl*-basiertes Paket, das Ihre DNS-Daten durchwandert und nach häufigen Fehlern sucht und sicherstellt, daß die Informationen konsistent sind. *dnswalk* wurde vor einiger Zeit in *comp.sources.misc* gepostet und sollte auf allen FTP-Sites verfügbar sein, die diese Gruppe archivieren. Es ist auch in den Quellen zu BIND-4.9.3 enthalten.

Fußnoten

(1)
: Die Funktionsweise von DNS wurde in Kapitel 2 beschrieben.

(2)
: NIS steht für Network Information System und wird in Kapitel 10 beschrieben.

(3)
: Eine detaillierte Diskussion dieses Problems finden Sie in RFC 1535.

(4)
: BIND-4.9.3 wird von Paul Vixie (paul@vix.com) entwickelt.

(5)
: Auf Linux-FTP-Sites liegen named-Binaries, die sich alle ein wenig in ihrer Konfiguration unterscheiden. Beispielsweise legen einige ihre PID-Datei in /etc ab, andere in /tmp usw.

(6)
: Beachten Sie, daß die Domain-Namen in dieser Datei ohne Punkt am Ende angegeben werden müssen. Frühere Versionen von named scheinen nachfolgende Punkte in named.boot als Fehler zu betrachten, und ignorieren die Zeile kommentarlos. Dies ist in BIND-4.9.3 behoben.

(7)
: Die aktuelle Implementation beschränkt die Liste auf 10 Einträge.

(8)
: Es gibt allerdings andere Methoden, die Namensliste einer Domain herauszubekommen, sie sind aber nicht so simpel wie ein Zonentransfer.

(9)
: Obwohl die Dokumentation besagt, daß dies für alle Records gilt, scheint dies nur beim SOA-Record der Fall zu sein und auch hier nur nach einem bestimmten Eintrag.

(10)
: Wie haben es hier leider mit einem Henne-und-Ei-Problem zu tun: Sie können Ihren Name-Server nicht nach den Root-Servern fragen, wenn Sie ihm noch keine mitgeteilt haben. Um aus diesem Dilemma herauszukommen, können Sie entweder nslookup dazu bewegen, einen anderen Name-Server zu verwenden, oder die Daten aus diesem Beispiel als Ausgangspunkt nehmen und damit die volle Liste der Root-Server erfragen.

(11)
: In älteren Versionen von BIND war es möglich, einen Eintrag für localhost. einfach in eine andere Zone »einzuschmuggeln«, z.B. in die Datei named.hosts. Seit Version 4.9.3 ignoriert named alle Einträge, die nicht zu der Domain gehören, die im SOA-Record angegeben ist.

Kapitel 7

Serial Line IP

Die Serial Line-Protokolle SLIP und PPP bieten Internet-Connectivity für Arme. Außer einem Modem und einer mit einem FIFO-Buffer ausgestatteten seriellen Schnittstelle wird keine weitere Hardware benötigt. Die Verwendung ist nicht schwieriger als bei einer Mailbox, und eine ständig steigende Zahl privater Organisationen und in letzter Zeit auch kommerzieller Provider bietet Dialup-IP zu bezahlbaren Preisen für jedermann an.

Für Linux stehen sowohl SLIP- als auch PPP-Treiber (siehe Kapitel 8, Das Point-to-Point-Protokoll) zur Verfügung. SLIP ist schon seit langem verfügbar, und ein PPP-Treiber wurde vor einiger Zeit von Michael Callahan und Al Longyear entwickelt. Beide laufen mittlerweile stabil.

Allgemeine Anforderungen

Um SLIP oder PPP zu verwenden, müssen Sie einige der grundlegenden Netzwerk-Funktionen installieren, die wir in den vorherigen Kapiteln besprochen haben. Als Minimum ist das Loopback-Interface einzurichten und die Auflösung von Namen zu ermöglichen. Wenn Sie sich an das Internet anbinden wollen, werden Sie natürlich auch DNS verwenden wollen. Die einfachste Möglichkeit besteht darin, die Adresse irgendeines Name-Servers in Ihre *resolv.conf*-Datei einzutragen. Dieser Server wird dann abgefragt, sobald der SLIP-Link aktiviert wird. Je näher (netzwerktechnisch) der Server an Ihrem Einwählpunkt liegt, desto besser werden die Antworten sein.

Allerdings ist diese Lösung nicht optimal, weil alle Lookups immer noch über Ihren SLIP/PPP-Link laufen. Wenn Sie sich über die auf diese Weise verbrauchte Bandbreite Sorgen machen, können Sie einen *Caching-Only*-Name-Server einrichten. Dieser bedient keine bestimmte Domain, sondern arbeitet als Relais für alle auf Ihrem Rechner produzierten DNS-Anfragen. Der Vorteil dieses Schemas liegt darin, daß ein Cache aufgebaut wird und die meisten Anfragen auf diese Weise nur einmal über die serielle Leitung laufen. Eine *named.boot*-Datei für einen Caching-Only-Server sieht ungefähr so aus:

```
; named.boot-Datei für Caching-Only-Server
directory                              /var/named
primary       0.0.127.in-addr.arpa   db.127.0.0 ; Loopback-Netz
```

```
cache          .                              db.cache   ; Root-Server
```

Neben der Datei *named.boot* müssen Sie auch eine Datei namens *db.cache* einrichten, die eine gültige Liste aller Root-Name-Server enthält. Wie das geht, wird am Ende des Kapitels 6, *Resolver und Name-Server* beschrieben.

SLIP-Betrieb

Dialup-IP-Server bieten SLIP häufig über spezielle Benutzer-Accounts an. Nach dem Einloggen in einen solchen Account landen Sie nicht in der üblichen Shell, sondern es wird ein Programm oder Shell-Script ausgeführt, das den SLIP-Treiber des Servers für die serielle Leitung konfiguriert und ein entsprechendes Netzwerk-Interface aufbaut. Dasselbe muß danach an Ihrem Ende der Leitung passieren.

Bei manchen Betriebssystemen ist der SLIP-Treiber ein normales Anwenderprogramm; bei Linux dagegen ist er Teil des Kernel und ist dadurch deutlich schneller. Das verlangt allerdings, daß die serielle Leitung explizit auf den SLIP-Modus umgestellt wird. Dieser Modus ist als eine spezielle »line discipline« (Betriebsmodus) implementiert. Wenn das Tty sich im Normalmodus (DISC0) befindet, tauscht es Daten nur mit Benutzerprozessen über die Systemaufrufe *read(2)*- und *write(2)* aus, und der SLIP-Treiber ist nicht in der Lage, von diesem Tty zu lesen oder darauf zu schreiben. Bei SLIPDISC sind die Rollen vertauscht: Nun werden alle Schreib- und Lesezugriffe von Benutzerprozessen blockiert, während alle über die serielle Schnittstelle eintreffenden Daten direkt an den SLIP-Treiber weitergeleitet werden.

Der SLIP-Treiber selbst versteht eine Reihe von Variationen des SLIP-Protokolls. Neben dem normalen SLIP versteht er auch CSLIP, das die sogenannte Van Jacobson-Header-Komprimierung von IP-Paketen implementiert. Das verbessert den Durchsatz bei interaktiven Anwendungen merklich. Darüber hinaus gibt es 6-Bit-Versionen dieser Protokolle.

Einen einfachen Weg der Umstellung einer seriellen Leitung auf den SLIP-Modus bietet das *slattach*-Tool. Nehmen wir einmal an, Sie hätten Ihr Modem an */dev/cua3* angeschlossen und sich erfolgreich in den SLIP-Server eingeloggt. Nun führen Sie den folgenden Befehl aus:

```
# slattach /dev/cua3 &
```

Das schaltet die Leitungsart von *cua3* auf SLIPDISC und verbindet diese mit einer der SLIP-Netzwerk-Schnittstellen. Ist dies Ihr erster aktiver SLIP-Link, wird die Leitung an *sl0* gebunden; die zweite an *sl1* und so weiter. Die maximale Anzahl unterstützter Schnittstellen ist von der Version und Konfiguration Ihres Kernels abhängig, im allgemeinen liegt der Wert bei 4 oder 8.

Per Voreinstellung verwendet *slattach* CSLIP. Sie können einen anderen Modus mit der Kommandozeilen-Option *-p* einstellen. Wenn Sie mit normalem SLIP (ohne Kompression) arbeiten wollten, würden Sie folgendes eingeben:

```
# slattach -p slip /dev/cua3 &
```

Andere Modi sind *cslip*, *slip6*, *cslip6* (für die 6-Bit-Versionen von SLIP) und *adaptive* für adaptives SLIP. Die letzte Variante überläßt es dem Kernel, herauszufinden, welchen SLIP-Modus das Gegenüber verwendet.

Beachten Sie, daß Sie (außer bei *adaptive*) dieselbe Variante verwenden müssen wie Ihre Gegenstelle. Wenn beispielsweise der Host **cowslip** CSLIP verwendet, müssen Sie dies auch tun. Ein typisches Symptom unterschiedlicher SLIP-Modi ist die Fehlermeldung »`Can't build ICMP header`«, die der Kernel ausspuckt, wenn Sie z. B. mit *telnet* auf einen entfernten Host zugreifen wollen. Es kann aber auch sein, daß Sie gar nichts sehen, und *telnet* einfach nur zu »hängen« scheint. Solchen Problemen können Sie beispielsweise durch die Verwendung von adaptivem SLIP aus dem Weg gehen.

Mit *slattach* können Sie nicht nur SLIP, sondern auch andere Protokolle wie PPP oder KISS (ein von Amateurfunkern verwendetes Protokoll) aktivieren, die serielle Leitungen verwenden. Details können Sie der *slattach(8)*-Manpage entnehmen.

Nachdem die Leitung nun für den SLIP-Treiber bereitsteht, müssen Sie das Netzwerk-Interface konfigurieren. Dies erledigen Sie wieder mit den Standardbefehlen *ifconfig* und *route*. Nehmen wir einmal an, daß Sie von **vlager** aus einen Server namens **cowslip** angewählt haben. Sie würden dann die folgenden Befehle eingeben:

```
# ifconfig sl0 vlager-slip pointopoint cowslip
# route add cowslip
# route add default gw cowslip
```

Der erste Befehl konfiguriert das Interface als Point-to-Point-Link zu cowslip, während der zweite und der dritte die Route zu cowslip und die Default-Route setzen, wobei cowslip als Gateway verwendet wird. gateway.

Zwei Dinge lohnen die besondere Beachtung bei diesem *ifconfig*-Aufruf. Das erste ist die Option `pointopoint`, mit der die Adresse des anderen Endes der Punkt-zu-Punkt-Verbindung festgelegt wird. Das zweite ist die Verwendung von **vlager-slip** als Adresse für das lokale SLIP-Interface.

Wir haben bereits vorher besprochen, daß Sie die Adresse, die dem Ethernet-Interface von **vlager** zugewiesen wurde, auch für Ihren SLIP-Link verwenden können. In diesem Fall muß **vlager-slip** ein weiteres Alias für die Adresse **172.16.1.1** sein. Andererseits können Sie Ihrem SLIP-Link aber auch eine ganz andere Adresse zuweisen. So ein Fall wäre beispielsweise gegeben, wenn Ihr Netzwerk eine nicht registrierte IP-Netzwerk-Adresse verwendet, wie das die Brauerei tut. Wir kommen auf dieses Thema im nächsten Abschnitt ausführlicher zurück.

Für den Rest dieses Kapitels werden wir immer **vlager-slip** verwenden, wenn wir uns auf die Adresse des lokalen SLIP-Interfaces beziehen.

Um den SLIP-Link herunterfahren, beenden Sie einfach den *slattach*-Prozeß, indem Sie ihm das Hangup-Signal schicken:

```
# kill -HUP 516
```

Handhabung privater IP-Netzwerke

Sie werden sich aus Kapitel 5 daran erinnern, daß die virtuelle Brauerei nichtregistrierte Netzwerknummern verwendet, die nur für den internen Gebrauch reserviert sind. Pakete von oder zu einem dieser Netzwerke werden vom Internet nicht geroutet. Das bedeutet, daß Hosts innerhalb des Brauerei-Netzwerks nicht mit echten Internet-Hosts kommunizieren können, weil die entsprechenden Pakete vom ersten größeren Router still und leise aussortiert werden.

Um dieses Problem zu umgehen, konfigurieren wir **vlager** so, daß er als eine Art Verteiler für Internet-Dienste agiert. Der restlichen Welt stellt er sich als normaler Internet-Host mit einer (üblicherweise vom Netzwerk-Provider zugewiesenen) registrierten Internet-Adresse dar. Um auf einen Internet-Host, beispielsweise einen FTP-Server, zugreifen zu können, müssen sich Benutzer in **vlager** einloggen und den FTP-Client von dort aus starten, so daß es so aussieht, als käme die Verbindung von einer gültigen Adresse. Bei anderen Anwendungen könnte es Lösungen geben, bei denen der Benutzer sich nicht bei **vlager** einloggen muß. Zum Beispiel könnten WWW-Benutzer einen sog. *Proxy Server* auf **vlager** starten, der alle Anforderungen der Benutzer an die entsprechenden Server weiterleitet.

Das ist natürlich etwas umständlich, aber neben der ganzen Schreibarbeit, die Sie sich sparen, weil Sie keine IP-Adressen registrieren müssen, hat dieses Vorgehen noch den zusätzlichen Vorteil, daß es sich ausgezeichnet in ein Firewall-Konzept einpaßt. Firewalls sind dedizierte Hosts, die verwendet werden, um Benutzern Ihres lokalen Netzwerks eingeschränkten Zugang zum Internet zu verschaffen, ohne die internen Hosts Netzwerk-Attacken von außen auszusetzen.

Nehmen wir einmal an, die Brauerei hat die IP-Adresse **192.168.5.74** für den SLIP-Zugriff vorgesehen. Alles was Sie nun tun müssen, um die oben diskutierte Konfiguration zu realisieren, ist, diese Adresse in die Datei */etc/hosts* einzutragen und sie **vlager-slip** zu nennen. Die Prozedur, wie der SLIP-Link hochgezogen wird, bleibt dabei unverändert.

Verwendung von dip

Nun, das war ja ziemlich einfach. Nichtsdestotrotz wollen Sie wahrscheinlich die obigen Schritte so automatisieren, daß Sie nur einen einzigen Befehl eingeben müssen, der dann all die oben aufgeführten Befehle durchführt. Genau das erledigt *dip*.

dip steht für *Dialup IP*. Geschrieben wurde es von Fred van Kempen. Als dieses Buch geschrieben wurde, war gerade die Version 3.3.7 aktuell. Das Programm wurde von einer ganzen Reihe von Leuten (z.T. sehr stark) gepatcht, so daß man nicht mehr von einem einzelnen *dip*-Programm sprechen kann. Die unterschiedlichen Entwicklungen werden hoffentlich in einer späteren Release zusammengefaßt.

dip bietet einen Interpreter für eine einfache Script-Sprache, mit der Sie Ihr Modem steuern, Leitungen in den SLIP-Modus schalten und die Schnittstellen konfigurieren können. Es ist recht einfach gehalten und eher restriktiv, reicht in den meisten Fällen aber aus. Neue Releases von *dip* könnten in Zukunft eine vielseitigere Sprache anbieten.



Um das SLIP-Interface konfigurieren zu können, benötigt *dip* Root-Privilegien. Es wäre nun sehr

verlockend, *dip* mit Setuid auf **root** zu setzen, so daß alle Benutzer einen SLIP-Server anwählen können, ohne Root-Rechte zu besitzen. Das ist aber sehr gefährlich, weil die fehlerhafte Einrichtung von Interfaces und Default-Routen mit *dip* das Routing auf Ihrem Netzwerk zum Erliegen bringen könnte. Was aber noch schlimmer ist, ist die Tatsache, daß Sie Ihren Benutzern auf diese Weise die Möglichkeit geben, sich an *jeden* SLIP-Server anzuschließen, und so gefährliche Angriffe auf Ihr Netzwerk provozieren könnten. Wenn Sie es Ihren Benutzern also erlauben wollen, SLIP-Verbindungen aufzubauen, sollten Sie kleine Programme für jeden vorgesehenen SLIP-Server schreiben, die *dip* dann mit dem entsprechenden Script aufrufen, das die Verbindung sauber aufbaut. Diese Programme können dann bedenkenlos mit Setuid auf **root** gesetzt werden.

Ein Beispiel-Script

Nehmen wir an, daß **cowslip** der Host ist, zu dem Sie die SLIP-Verbindung aufbauen wollen, und daß Sie für *dip* ein Skript namens *cowslip.dip* geschrieben haben, das die Verbindung herstellt. Wir starten *dip* und übergeben den Script-Namen als Argument:

```
# dip cowslip.dip
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.
connected to cowslip.moo.com with addr 192.168.5.74
#
```

Das Script selbst ist in Beispiel 7--1 zu sehen.

Beispiel 7-1. dip-Beispiel-Script

```
#dip-Beispiel-Script zur Anwahl von cowslip
# setze lokale und entfernte Namen/Adressen
get $local vlager-slip
get $remote cowslip
port cua3                    # wähle seriellen Port
speed 38400                  # Geschwindigkeit auf Maximum
modem HAYES                  # setze Modemtyp
reset                        # Modem und TTY rücksetzen
flush                        # Eingabepuffer leeren
# Wahl vorbereiten
send ATQ0V1E1X1\r
wait OK 2
if $errlvl != 0 goto error
dial 41988
if $errlvl != 0 goto error
wait CONNECT 60
if $errlvl != 0 goto error
# nun sind wir verbunden
sleep 3
send \r\n\r\n
```

```
wait ogin: 10
if $errlvl != 0 goto error
send Svlager\n
wait ssword: 5
if $errlvl != 0 goto error
send hey-joe\n
wait running 30
if $errlvl != 0 goto error
# wir sind eingeloggt, und die andere Seite setzt SLIP auf
print Connected to $remote with address $rmtip
default                  # dieser Link ist die Default-Route
mode SLIP                # wir wechseln nun auch in den SLIP-Modus
# im Fehlerfall hierhin verzweigen
error:
print SLIP to $remote failed.
```

Nachdem die Verbindung zu **cowslip** steht und SLIP aktiviert ist, trennt sich *dip* vom Terminal ab und geht in den Hintergrund. Sie können dann die normalen Netzwerkdienste über den SLIP-Link benutzen. Um die Verbindung zu beenden, müssen Sie den *dip*-Befehl mit der Kommandozeilen-Option *-k* aufrufen. Damit wird ein Hangup-Signal an *dip* gesendet, wobei die Prozeß-ID verwendet wird, den *dip* in der Datei */etc/dip.pid* eingetragen hat:

```
# dip -k
```

Bei der von *dip* zur Verfügung gestellten Script-Sprache bezeichnen Schlüsselworte, denen ein Dollarzeichen vorangestellt ist, immer einen Variablennamen. *dip* besitzt eine ganze Reihe vordefinierter Variablen, die nachfolgend noch aufgeführt werden. Beispielsweise enthalten `$remote` und `$local` die Hostnamen der für den SLIP-Link verwendeten entfernten bzw. lokalen Hosts.

Bei den ersten beiden Befehlszeilen im Beispiel-Script werden *get*-Befehle verwendet, mit denen bei *dip* Variablen gesetzt werden. In diesem Fall wird der entfernte und der lokale Hostname auf **vlager** bzw. **cowslip** gesetzt.

In den nächsten fünf Befehlszeilen werden die Terminalleitung und das Terminal eingerichtet. `reset` sendet einen Reset-String an das Modem (bei Hayes-kompatiblen Modems ist dies der `ATZ`-Befehl). Der nächste Befehl leert den Eingabepuffer, so daß der Login-Dialog korrekt funktionieren kann. Dieser Dialog geht ist ziemlich simpel: *dip* wählt einfach 41988, die Telefonnummer von **cowslip**, und loggt sich unter dem Account `Svlager` mit dem Paßwort `hey-joe` ein. Der `wait`-Befehl veranlaßt *dip*, auf die Zeichenkette zu warten, die als erstes Argument übergeben wurde. Das zweite Argument bestimmt die Zeit, die auf diese Zeichenkette gewartet werden soll. Die in der Login-Prozedur eingestreuten `if`-Befehle prüfen, ob während der Ausführung irgendwelche Fehler aufgetreten sind.

Die nach dem Einloggen noch verbleibenden Befehle sind `default`, der den SLIP-Link als Default-Route für alle Hosts definiert, und `mode`, der den SLIP-Modus für diese Leitung aktiviert und das Interface und die Routing-Tabelle für Sie konfiguriert.

dip-Referenz

Obwohl es so weit verbreitet ist, ist *dip* bisher nicht besonders gut dokumentiert. Dieser Abschnitt enthält daher eine Referenz zu den meisten *dip*-Befehlen. Sie erhalten eine Übersicht aller Befehle, indem Sie *dip* im Testmodus starten und dann den *help*-Befehl eingeben. Um etwas zur Syntax eines Befehls zu erfahren, geben Sie den Befehl einfach ohne weitere Argumente ein (was natürlich bei Befehlen, die keine zusätzlichen Argumente benötigen, nicht funktioniert).

```
$ dip -t
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.
DIP> help
DIP knows about the following commands:
  databits default  dial     echo     flush
  get      goto     help     if       init
  mode     modem    parity   print    port
  reset    send     sleep    speed    stopbits
  term     wait
DIP> echo
Usage: echo on|off
DIP> _
```

Im gesamten folgenden Abschnitt zeigen die Beispiele, bei denen der *DIP>*-Prompt auftaucht, wie Sie einen Befehl im Testmodus eingeben müssen, und welche Ausgabe dieser Befehl dann erzeugt. Fehlt in Beispielen dieser Prompt, sollten Sie sie als Auszüge von Scripten betrachten.

Modembefehle

dip stellt eine ganze Reihe von Befehlen zur Verfügung, mit denen Sie Ihre serielle Leitung und Ihr Modem konfigurieren können. Die Funktion ist bei einigen Befehlen offensichtlich; beispielsweise wählt `port` den verwendeten seriellen Port an. Weitere solche Befehle sind `speed`, `databits`, `stopbits` und `parity`, mit denen Sie die gängigen Parameter der Leitung einstellen können.

Mit dem `modem`-Befehl können Sie den aktuellen Modemtyp wählen. Momentan ist `HAYES` (in Großbuchstaben!) der einzig unterstützte Typ. Sie müssen *dip* den Modemtyp bekanntgeben, weil es sonst die Befehle `dial` und `reset` nicht fehlerfrei ausführen kann. Der Befehl `reset` sendet einen Reset-String an das Modem. Dieser String ist vom verwendeten Modem abhängig. Bei Hayes-kompatiblen Modems lautet dieser String `ATZ`.

Mit dem `flush`-Befehl werden alle noch im Eingabepuffer befindlichen Antworten des Modems weggeworfen. Anderenfalls könnte das einem `reset` folgende Dialog-Script verwirrt sein, weil es die `OK`s früherer Befehle vorfinden würde.

Mit dem `init`-Befehl wählen Sie einen Initialisierungs-String, der an das Modem geschickt wird, bevor gewählt wird. Die Standardeinstellung für Hayes-Modems lautet »`ATE0 Q0 V1 X1`«, womit die Ausgabe (Echo) von Befehlen und langen Ergebniskodes unterdrückt wird und die »Blindwahl« (keine Prüfung des Wähltons) aktiviert wird.

Schließlich sendet der `dial`-Befehl einen Initialisierungs-String an das Modem und wählt den anderen Rechner an. Der Standard-Wählbefehl für Hayes-Modems lautet `ATD`.

echo und term

Der `echo`-Befehl dient als Debugging-Hilfe. Mit `echo on` werden von *dip* alle an das serielle Gerät geschickten Daten auch auf der Konsole ausgegeben. Diese Option können Sie mit dem Befehl `echo off` wieder abschalten.

dip erlaubt Ihnen, den Script-Modus kurzfristig zu verlassen und den Terminalmodus zu verwenden. In diesem Modus arbeitet *dip* wie ein normales Terminalprogramm, das Daten an die serielle Leitung schickt und von ihr liest. Mit Ctrl-] setzen Sie die Abarbeitung der Skripten fort.

Der get-Befehl

Mit dem Befehl `get` können Sie Variablen für *dip* setzen. Die einfachste Form ist, eine Variable auf einen konstanten Wert zu setzen, wie wir das bei *cowslip.dip* getan haben. Darüber hinaus können Sie aber auch Eingaben vom Benutzer anfordern, indem Sie das Schlüsselwort `ask` anstelle eines Wertes angeben:

```
DIP> get $local ask
Enter the value for $local: _
```

Eine dritte Möglichkeit ist, zu versuchen, den Wert vom entfernten Host zu erhalten. So seltsam das auf den ersten Blick auch erscheinen mag, so nützlich ist dies in manchen Fällen. Manche SLIP-Server erlauben Ihnen nicht die Verwendung einer eigenen IP-Adresse für den SLIP-Link, sondern weisen Ihnen beim Einwählen eine Adresse aus einem Adreß-Pool zu, wobei Sie eine Meldung erhalten, welche Adresse Ihnen zugewiesen wurde. Wenn die Nachricht in etwa die folgende Form hat `Your address: 192.168.5.74`, können Sie mit den folgenden Befehlszeilen Ihre Adresse ermitteln:

```
# finish login
wait address: 10
get $locip remote
```

Der print-Befehl

Mit diesem Befehl können Sie Texte auf die Konsole ausgeben, von der aus *dip* gestartet wurde. Auch die von *dip* verwendeten Variablen können in print-Befehlen verwendet werden:

```
DIP> print Using port $port at speed $speed
Using port cua3 at speed 38400
```

Variablennamen

dip versteht nur eine Reihe vordefinierter Variablen. Ein Variablenname beginnt immer mit einem Dollarzeichen und muß in Kleinbuchstaben geschrieben werden.

Die Variablen `$local` und `$locip` enthalten den lokalen Hostnamen und die IP-Adresse. Durch Setzen des Hostnamens werden der kanonische Hostname in `$local` und gleichzeitig die entsprechende Adresse in `$locip` gespeichert. Analog geht es auch beim Setzen von `$locip` vonstatten.

Die Variablen `$remote` und `$rmtip` speichern die entsprechenden Angaben für den entfernten Host. `$mtu` enthält den MTU-Wert dieser Verbindung.

Diese fünf Variablen sind die einzigen, denen Sie mit `get` direkt Werte zuweisen können. Eine Menge anderer Variablen kann nur über spezielle Befehle gesetzt, aber über `print`-Befehle ausgegeben werden: `$modem`, `$port` und `$speed`.

`$errlvl` ist die Variable, mit der Sie auf das Resultat des zuletzt ausgeführten Befehls zugreifen können. Ist der Wert null, ist die Operation erfolgreich durchgeführt worden, bei einem Wert ungleich null ist ein Fehler aufgetreten.

Die if- und goto-Befehle

Der Befehl `if` ist sehr einfach gehalten und erlangt nur eine bedingte Verzweigung an eine andere Stelle des Skripts. Die Syntax lautet:

```
if var op number goto label
```

Der Ausdruck darf nur ein einfacher Vergleich zwischen einer der Variablen `$errlvl`, `$locip`, `$rmtip` und einer ganzen Zahl sein. Der Operator *op* kann ==, !=, <, >, <= oder >= sein.

Mit dem `goto`-Befehl wird die Ausführung des Scripts in der Zeile fortgesetzt, die der das *label* enthaltenden Zeile folgt. Ein Label muß an der ersten Stelle einer Zeile beginnen, und es muß ihm unmittelbar ein Doppelpunkt folgen.

send, wait und sleep

Diese Befehle helfen bei der Implementierung einfacher Dialog-Scripten unter *dip*. `send` überträgt seine Argumente an die serielle Leitung. Variablen werden nicht unterstützt, wohl aber C-ähnliche Backslash-Sequenzen wie `\n` und `\b`. Das Tildezeichen (~) wird als Abkürzung für die Zeichenkombination Wagenrücklauf/Zeilenvorschub verwendet.

`wait` akzeptiert ein Wort als Argument und liest die Eingaben von der seriellen Schnittstelle, bis es dieses Wort findet. Das Wort selbst darf keine Leerzeichen enthalten. Optional können Sie `wait` einen Timeout-Wert (in Sekunden) als zweites Argument übergeben. Wird das gewünschte Wort nicht innerhalb dieser Zeit gefunden, bricht der Befehl ab und setzt die Variable `$errlvl` auf eins.

Mit dem `sleep`-Befehl können Sie die weitere Befehlsausführung für eine bestimmte Zeit anhalten, beispielsweise um das Ende einer Login-Sequenz abzuwarten. Auch bei diesem Befehl wird die zu wartende Zeit in Sekunden angegeben.

mode und default

Diese Befehle werden verwendet, um die serielle Leitung in den SLIP-Modus zu schalten und das Interface zu konfigurieren.

`mode` ist der letzte Befehl, der von *dip* ausgeführt wird, bevor es in den Dämon-Modus wechselt. Wenn kein Fehler auftritt, kehrt der Befehl nicht zurück.

`mode` erwartet einen Protokollnamen als Argument. *dip* akzeptiert momentan `SLIP` und `CSLIP` als gültige Werte. Leider kann die derzeitige Version von *dip* noch nicht mit adaptivem SLIP umgehen. Nachdem der SLIP-Modus für die serielle Leitung aufgesetzt wurde, führt *dip* den Befehl *ifconfig* aus, um das Interface zu konfigurieren. Danach wird noch *route* gestartet, um die Route zum entfernten Host einzurichten.

Wenn das Script vor dem `mode`-Befehl noch den `default`-Befehl ausführt, dann sorgt *dip* dafür, daß zusätzlich die Default-Route auch auf diesen SLIP-Link zeigt.

Der Server-Modus

Den SLIP-Client einzurichten war der schwierige Teil. Das Gegenteil davon, nämlich Ihren Host so zu konfigurieren, daß er als SLIP-Server arbeitet, ist wesentlich einfacher.

Eine Möglichkeit wäre, *dip* im Server-Modus zu betreiben, was Sie einfach durch die Eingabe von *diplogin* erreichen. Die Haupt-Konfigurationsdatei ist */etc/diphosts*, bei der jeder Loginname mit der Adresse verknüpft ist, der der Host zugewiesen ist. Alternativ können Sie *sliplogin* verwenden, ein BSD-basiertes Tool, das ein wesentlich flexibleres Konfigurationsschema unterstützt. Dank dieses Schemas können Sie beliebige Shell-Scripten ausführen, wenn ein Host die Verbindung aufbaut oder beendet. Das Programm befindet sich gerade im Betatest.

Beide Programme erwarten, daß Sie einen Login-Account pro SLIP-Client einrichten. Nehmen wir mal an, Sie wollen Arthur Dent, der an **dent.beta.com** sitzt, mit SLIP-Diensten versorgen. Sie können einen Account namens **dent**einrichten, indem Sie die folgende Zeile in Ihre *passwd*-Datei aufnehmen:

```
dent:*:501:60:Arthur Dent's SLIP account:/tmp:/usr/sbin/diplogin
```

Das Paßwort für **dent**würden Sie danach mit dem *passwd*-Utility einrichten.

Wenn sich **dent** nun einloggt, fährt *dip* als Server hoch. Um zu ermitteln, ob er wirklich berechtigt ist, SLIP zu verwenden, sucht das Programm den Benutzernamen in der Datei */etc/diphosts*. Diese Datei enthält Angaben zu den Zugriffsrechten und Verbindungsparametern jedes SLIP-Benutzers. Beispielsweise könnte der Eintrag für **dent** wie folgt aussehen:

```
dent::dent.beta.com:Arthur Dent:SLIP,296
```

Das erste durch Doppelpunkte getrennte Feld enthält den Namen, unter dem sich der Benutzer einloggen muß. Das zweite Feld kann ein zusätzliches Paßwort enthalten (siehe unten). Im dritten Feld steht der Hostname oder die IP-Adresse des anrufenden Host. Als nächstes kommt ein Feld ohne besondere Bedeutung (noch). Das letzte Feld beschreibt die Verbindungsparameter. Dabei handelt es sich um eine durch Kommata getrennte Liste, die das verwendete Protokoll (momentan `SLIP` oder `CSLIP`) und die MTU enthält.

Loggt sich **dent** ein, extrahiert *diplogin* die erforderlichen Informationen über ihn aus der Datei *diphosts*. Ist das zweite Feld dabei nicht leer, wird nach einem »externen Sicherheitspaßwort.« gefragt. Der vom

Benutzer eingegebene String wird mit dem (unverschlüsselten) Paßwort aus *diphosts* verglichen. Stimmen beide nicht überein, wird das Login verweigert.

Sonst schaltet *diplogin* die serielle Leitung in den CSLIP- oder SLIP-Modus und richtet das Interface und die Route ein. Die Verbindung wird so lange aufrechterhalten, bis der Benutzer seine Arbeit beendet, und das Modem die Leitung unterbricht. *diplogin* setzt die Leitungsparameter dann wieder zurück und beendet seine Operation.

diplogin benötigt Superuser-Privilegien. Läuft *dip* ohne Setuid **root**, sollte *diplogin* eine separate Kopie von *dip* erhalten und nicht nur einen einfachen Link. Auf *diplogin* kann dann ohne Bedenken Setuid angewandt werden, ohne den Status von *dip* selbst zu beeinträchtigen.

Kapitel 8
Das Point-to-Point-Protokoll

Die Ps entwirren

Genau wie SLIP ist auch PPP ein Protokoll, mit dem Sie Datagramme über eine serielle Leitung übertragen können, das aber eine Reihe von Defiziten von SLIP behebt. Es erlaubt den kommunizierenden Rechnern, sich über Optionen wie IP-Adressen und die maximale Datagramm-Größe während der Startphase zu verständigen und stellt Mechanismen zur Autorisierung des kommunizierenden Partners bereit. Für jede dieser Fähigkeiten verwendet PPP ein separates Protokoll. In diesem Kapitel behandeln wir die grundlegenden Bausteine von PPP. Die Diskussion ist weit davon entfernt, vollständig zu sein. Wenn Sie mehr über PPP wissen wollen, sollten Sie sich die entsprechende RFC-Spezifikation sowie das gute Dutzend begleitender RFCs ansehen.(1)

Die Basis bei PPP bildet das *High-Level Data Link Control*-Protokoll, oder kurz HDLC.(2) Dieses Protokoll definiert die Begrenzung der einzelnen PPP-Frames und liefert eine 16-Bit-Prüfsumme. Im Gegensatz zur eher primitiven SLIP-Kapselung ist ein PPP-Frame in der Lage, auch Pakete anderer Protokolle als IP zu transportieren (z. B. Novells IPX oder AppleTalk). PPP erreicht dies, indem es ein Protokollfeld in den HDLC-Frame aufnimmt, der den Typ des Pakets bestimmt, das sich in diesem Frame befindet.

LCP, das *Link Control Protocol*, sitzt über HDLC und dient dazu, die Konfiguration der Leitung zwischen beiden Partnern auszuhandeln. Dazu gehört beispielsweise die *Maximum Receive Unit* (MRU), mit der die maximale Datagramm-Größe bestimmt wird, die ein Host zu empfangen bereit ist.

Ein wichtiger Schritt bei der Konfigurationsphase eines PPP-Links ist die Client-Authentisierung. Obwohl es nicht vorgeschrieben ist, ist es doch ein Muß bei Wählverbindungen. Üblicherweise fordert der angerufene Host (der Server) den Client auf, sich auszuweisen, indem er beweist, daß er einen geheimen Schlüssel (z. B. ein Paßwort) kennt. Wenn der Anrufer nicht mit der richtigen Antwort aufwarten kann, wird die Verbindung unterbrochen. Bei PPP funktioniert die Authentisierung in beiden Richtungen, d. h. der Anrufer kann auch den Server auffordern, sich zu authentisieren. Diese Authentisierungs-Prozeduren laufen völlig unabhängig voneinander ab. Es gibt zwei Protokolle für unterschiedliche Arten der Authentisierung, die wir weiter unten noch besprechen werden. Diese Protokolle heißen *Password Authentication Protocol* (PAP) und *Challenge Handshake Authentication Protocol* (CHAP).

Jedes Netzwerk-Protokoll wie IP, AppleTalk etc., das über die Datenverbindung geroutet wird, wird mit Hilfe eines entsprechenden *Network Control Protocol* (NCP) dynamisch konfiguriert. Sollen zum Beispiel IP-Datagramme über einen Link geschickt werden, müssen beide PPPs zuerst ermitteln, welche IP-Adresse von der jeweils anderen Seite verwendet wird. Das dabei benutzte Protokoll ist IPCP, das *Internet Protocol Control Protocol*.

Neben der Übertragung normaler IP-Datagramme unterstützt PPP auch die Header-Komprimierung von IP-Datagrammen nach Van Jacobson. Bei dieser Technik werden Header von TCP-Paketen auf eine Größe von bis zu drei Byte reduziert. Sie wird auch bei CSLIP verwendet und wird allgemein als VJ-Header-Komprimierung bezeichnet. Die Verwendung der Komprimierung kann ebenfalls während des Starts über IPCP vereinbart werden.

PPP auf Linux

Bei Linux ist die PPP-Funktionalität in zwei Teile unterteilt: in einen Low-Level-HDLC-Treiber, der im Kernel angesiedelt ist, und einen Dämon (*pppd*), der die verschiedenen Kontrollprotokolle behandelt. Das aktuelle Release von PPP für Linux ist *linux-ppp-2.1.2*. Sie besteht aus dem PPP-Kernel-Modul *pppd* und einem Programm namens *chat*, mit dem der entfernte Rechner angewählt wird.

Der PPP-Kernel-Treiber wurde von Michael Callahan geschrieben. *pppd* stammt ab von einer freien PPP-Implementierung für Sun- und 386BSD-Maschinen, die von Drew Perkins und anderen geschrieben und von Paul Mackerras gepflegt wurde. Auf Linux wurde sie von Al Longyear portiert.(3) *chat* wurde von Karl Fox geschrieben.(4)

Genau wie SLIP wird auch PPP über einen speziellen tty-Modus (line discipline) implementiert. Um eine Leitung als PPP-Link zu verwenden, bauen Sie die Verbindung wie gewohnt über Ihr Modem auf und schalten sie danach in den PPP-Modus. In diesem Modus werden alle eingehenden Daten an den PPP-Treiber weitergeleitet, der die eingehenden HDLC-Frames auf Gültigkeit prüft (jeder HDLC-Frame enthält eine 16-Bit-Prüfsumme), sie auspackt und verteilt. Im Moment können IP-Datagramme mit oder ohne Van Jacobson-Komprimierung verarbeitet werden. Sobald Linux IPX unterstützt, wird der PPP-Treiber auch für die Verarbeitung von IPX-Paketen erweitert.

Der Kernel-Treiber wird von dem Dämon *pppd* unterstützt, der die gesamte Initialisierungs- und Authentizierungsphase übernimmt, die notwendig ist, bevor die eigentlichen Daten über den Link geschickt werden können. Das Verhalten von *pppd* kann über eine ganze Reihe von Optionen getrimmt werden. Da PPP ziemlich komplex ist, ist es unmöglich, sie alle in einem einzigen Kapitel zu behandeln. Aus diesem Grund kann das Buch nicht alle Aspekte von *pppd* behandeln, sondern nur eine Einführung bieten. Für weitere Informationen seien Sie auf die Manpages und die *README*s der *pppd*-Source-Distribution verwiesen. Dort sollten Sie Antworten auf die meisten Fragen finden, die in diesem Kapitel nicht behandelt werden konnten. Wenn Sie nach dem Lesen der Dokumentation immer noch Probleme haben, sollten Sie sich an die Newsgruppe *comp.protocols.ppp* wenden. In dieser Newsgruppe finden Sie die meisten der Leute, die an der Entwicklung von *pppd* beteiligt waren.

Arbeiten mit pppd

Wenn Sie sich über einen PPP-Link an das Internet anschließen wollen, müssen Sie die grundlegenden Netzwerk-Fähigkeiten wie das Loopback Device und den Resolver eingerichtet haben. Wie das funktioniert, wurde in den vorhergehenden Kapiteln besprochen. Bei der Verwendung von DNS über serielle Leitungen kommen noch weitere Faktoren ins Spiel; entsprechende Hinweise finden Sie in Kapitel 7, *Serial Line IP*.

Um sich zur Einleitung vor Augen zu führen, wie eine PPP-Verbindung mit *pppd* aufgebaut wird, stellen Sie

sich vor, daß Sie wieder an **vlager** sitzen. Sie haben bereits den PPP-Server **c3po** angewählt und sich in den ppp-Account eingeloggt. **c3po** hat seinen PPP-Treiber bereits gestartet. Nachdem Sie das Kommunikationsprogramm beendet haben, mit dem Sie die Verbindung hergestellt hatten, führen Sie den folgenden Befehl aus:

```
# pppd /dev/cua3 38400 crtscts defaultroute
```

Mit diesem Befehl schalten Sie die serielle Leitung *cua3* in den PPP-Modus und bauen einen IP-Link zu **c3po** auf. Die Übertragungsgeschwindigkeit des seriellen Ports liegt bei 38400 bps. Mit der Option `crtscts` aktivieren Sie den Hardware-Handshake für diesen Port, was bei Geschwindigkeiten über 9600 bps ein absolutes Muß ist.

Nachdem *pppd* gestartet wurde, vereinbart es zuerst mit Hilfe von LCP verschiedene Verbindungseigenschaften mit dem anderen Ende. Üblicherweise funktionieren die Standardoptionen, die *pppd* auszuhandeln versucht, auf Anhieb, weshalb wir an dieser Stelle nicht weiter darauf eingehen.

Im Moment gehen wir auch davon aus, daß **c3po** von uns keine Authentisierung erwartet, so daß die Konfigurationsphase an dieser Stelle erfolgreich abgeschlossen ist.

pppd vereinbart dann mit Hilfe von IPCP, dem IP-Kontroll-Protokoll, die IP-Parameter mit seinem Gegenüber. Weil wir in der Befehlszeile keine bestimmte IP-Adresse an *pppd* übergeben haben, wird es versuchen, die Adresse zu verwenden, die der Resolver für den lokalen Hostnamen ermittelt hat. Beide geben sich dann ihre Adressen gegenseitig bekannt.

Üblicherweise gibt es mit diesen Voreinstellungen keine Schwierigkeiten. Selbst wenn Ihr Rechner in einem Ethernet hängt, können Sie dieselbe IP-Adresse sowohl für das Ethernet als auch für das PPP-Interface benutzen. Trotzdem erlaubt es Ihnen *pppd*, eine andere Adresse zu benutzen. Sie können sogar die Gegenstelle auffordern, eine bestimmte Adresse zu verwenden. Diese Optionen werden alle im Abschnitt »IP-Konfigurations-Optionen« behandelt.

Nachdem *pppd* die IPCP-Setup-Phase hinter sich gebracht hat, bereitet es die Netzwerkschicht Ihres Host erstmal für die Verwendung des PPP-Links vor. Zuerst konfiguriert es das PPP-Netzwerk-Interface als Point-to-Point-Link. Dabei verwendet es *ppp0* für den ersten aktiven PPP-Link, *ppp1* für den zweiten und so weiter. Danach richtet es einen Eintrag in der Routing-Tabelle ein, der auf den Host am anderen Ende des Links zeigt. In unserem obigen Beispiel läßt *pppd* die Default-Netzwerkroute auf **c3po** zeigen, weil wir die Option `defaultroute` verwendet haben.(5) Auf diese Weise werden alle Datagramme, die nicht für Hosts in Ihrem lokalen Netzwerk bestimmt sind, an **c3po** übertragen. *pppd* unterstützt eine Reihe unterschiedlicher Routing-Schemata, auf die wir später in diesem Kapitel noch eingehen werden.

Arbeiten mit Optionsdateien

Bevor *pppd* seine Kommandozeilen-Argumente bearbeitet, durchsucht es verschiedene Dateien nach Standardoptionen. Diese Dateien dürfen beliebige gültige Kommandozeilen-Argumente enthalten, die auch über mehrere Zeilen verteilt sein können. Kommentare werden dabei durch Doppelkreuze eingeleitet.

Die erste Optionsdatei ist */etc/ppp/options*, die immer durchsucht wird, während *pppd* startet. Es ist eine gute Idee, diese Datei für allgemeine Standardeinstellungen zu verwenden, weil Sie auf diese Weise verhindern, daß Benutzer verschiedene Dinge tun könnten, die sich auf die Systemsicherheit auswirken. Damit *pppd* beispielsweise irgendeine Form der Authentisierung (PAP oder CHAP) von der anderen Seite erwartet, könnten Sie die Option *auth* in diese Datei aufnehmen. Diese Option kann von einem Benutzer nicht

überschrieben werden, d. h. PPP kann keine Verbindung zu einem System aufbauen, das nicht in Ihrer Authentisierungs-Datei steht.

Nach */etc/ppp/options* wird die Optionsdatei *.ppprc* im Home-Verzeichnis des Benutzers gelesen. Sie erlaubt jedem Benutzer einen individuellen Satz von Standardoptionen.

Die Datei */etc/ppp/options* könnte beispielsweise wie folgt aussehen:

```
# Globale Optionen für pppd auf vlager.vbrew.com
auth                    # Authentisierung notwendig
usehostname             # verwende lokalen Hostnamen für CHAP
lock                    # verwende UUCP-konformes Device Locking
domain vbrew.com        # unser Domainname
```

Die ersten beiden Optionen dienen der Authentizierung und werden nachfolgend besprochen. Wegen des Schlüsselworts `lock` verwendet *pppd* die UUCP-Standardmethode zum Sperren der Gerätedateien. Nach dieser Konvention erzeugt jeder Prozeß, der auf ein serielles Device, beispielsweise */dev/cua3*, zugreift, eine Lock-Datei namens *LCK..cua3* im UUCP-Spool-Verzeichnis, um zu signalisieren, daß dieses Gerät benutzt wird. Ein solches Vorgehen ist notwendig, um sicherzustellen, daß andere Programme wie *minicom* oder *uucico* nicht auf das Gerät zugreifen, solange es von PPP verwendet wird.

Der Grund dafür, solche Optionen in die globale Konfigurationsdatei einzutragen, liegt darin, daß solche Optionen nicht überschrieben werden können und Sie so für einen vernünftigen Grad an Sicherheit sorgen. Andererseits existieren aber auch Optionen wie der `connect`-String, die auch später noch überschrieben werden können.

Anwählen mit Chat

Eine Sache, die Sie in unserem obigen Beispiel vielleicht als unbequem betrachten könnten, ist, daß Sie die Verbindung zuerst manuell aufbauen müssen, bevor Sie *pppd* starten können. Leider besitzt *pppd* im Gegensatz zu *dip* keine eigene Skript-Sprache, mit der Sie das entfernte System anwählen und sich einloggen können, sondern ist von einem externen Programm oder Shell-Skript abhängig, das diese Aufgabe übernimmt. Der entsprechende Befehl kann *pppd* mit der Kommandozeilen-Option `connect` übergeben werden. *pppd* leitet die Standard-Ein- und Ausgabe des Programms dann an die serielle Leitung um. Ein für diese Aufgabe ausgezeichnet geeignetes Programm ist *expect*, geschrieben von Don Libes. Es besitzt eine sehr mächtige, auf Tcl basierende Sprache und wurde genau für solche Anwendungszwecke entworfen.

Das *pppd*-Paket wird mit einem ähnlichen Programm namens *chat* geliefert, mit dem Sie Chat-Skripten ausführen können, wie sie auch von UUCP verwendet werden. Grundsätzlich besteht ein Chat-Skript aus einer abwechselnden Folge von Zeichenketten, die wir vom entfernten System erwarten, sowie den Antworten, die auf diese Strings übertragen werden sollen. Wir nennen sie *expect*- (die erwartete Zeichenkette) und *send*- (die zu übertragende Zeichenkette) Strings. Nachfolgend ein typischer Ausschnitt aus einem Chat-Skript:

```
ogin: b1ff ssword: s3kr3t
```

Diese Zeile weist *chat* an, darauf zu warten, bis die andere Seite den Login-Prompt gesendet hat, und dann den Loginnamen b1ff zurückzuschicken. Wir warten nur auf `ogin:`, so daß es keine Rolle spielt, ob der Login-Prompt mit einem großen oder kleinen »l« beginnt. Der folgende String ist ein weiterer erwarteter Text, auf den *chat* warten soll. Sobald dieser eingeht, wird unser Paßwort als Antwort übertragen.

Das ist im Grunde schon alles, was es zu Chat-Skripten zu sagen gibt. Ein vollständiges Skript, mit dem Sie einen PPP-Server anwählen würden, müßte natürlich auch die entsprechenden Modembefehle beinhalten. Nehmen wir einmal an, Ihr Modem verstünde den Hayes-Befehlssatz, und die Telefonnummer des Servers lautete 318714. Die vollständige Befehlszeile, mit der *chat* die Verbindung zu **c3po** aufbaut, würde dann lauten:

```
$ chat -v '' ATZ OK ATDT318714 CONNECT '' ogin: ppp word: GaGariN
```

Per Definition wird zuerst ein Expect-String erwartet, aber weil das Modem nichts sagen würde, wenn wir nicht ein wenig nachhelfen würden, veranlassen wir *chat*, den ersten Expect-String zu überspringen, indem wie einen leeren String angeben. Wir machen weiter und senden `ATZ`, den Reset-Befehl für Hayes-kompatible Modems, und warten auf die entsprechende Antwort (`OK`). Der nächste String sendet den Wählbefehl zusammen mit der gewünschten Telefonnummer an *chat* und erwartet die Nachricht `CONNECT` als Antwort. Dem folgt wieder ein leerer String, weil wir nun nichts mehr senden wollen, sondern nur noch auf den Login-Prompt warten. Der Rest des Skripts arbeitet genauso, wie bereits oben beschrieben.

Mit der Option *-v* weisen Sie *chat* an, alle Aktivitäten in der `local2`-Einrichtung des *syslog*-Dämons aufzuzeichnen.(6)

Die Angabe des Chat-Skripts auf der Kommandozeile birgt ein gewisses Risiko, weil Benutzer sich die Kommandozeilen laufender Prozesse mit dem *ps*-Befehl ansehen können. Sie können dieses Risiko ausschalten, indem Sie das Skript in einer Datei, z. B. *dial-c3po* speichern. Mit der Option *-f* (gefolgt vom Dateinamen) können Sie *chat* dann anweisen, das Skript aus dieser Datei zu lesen. Der vollständige *pppd*-Aufruf würde dann wie folgt lauten:

```
# pppd connect "chat -f dial-c3po" /dev/cua3 38400 -detach \
        crtscts modem defaultroute
```

Neben der Option *connect*, mit der das Skript bestimmt wird, haben wir noch zwei weitere Optionen in die Kommandozeile aufgenommen: zum einen *-detach*, mit der *pppd* angewiesen wird, sich nicht von der Konsole zu trennen und ein Hintergrundprozeß zu werden, zum anderen `modem`, wodurch modemspezifische Aktionen auf dem seriellen Gerät durchgeführt werden, beispielsweise die Unterbrechung der Leitung vor und nach jedem Anruf. Wenn Sie diese Option nicht verwenden, überwacht *pppd* nicht die DCD-Leitung des Ports und kann dann nicht erkennen, wenn das Gegenüber unvermittelt die Verbindung unterbricht.

Die obigen Beispiele waren ziemlich einfach; mit *chat* können Sie auch wesentlich kompliziertere Skripten erstellen. Eine sehr nützliche Möglichkeit ist, einen String angeben zu können, bei dem mit einer Fehlermeldung abgebrochen wird. Typische Strings sind Meldungen wie `BUSY` oder `NO CARRIER`, die normalerweise von Ihrem Modem erzeugt werden, wenn die gewählte Nummer besetzt ist oder nicht antwortet. Damit *chat* diese Nachrichten direkt erkennt und nicht erst auf den Timeout wartet, können Sie sie zusammen mit dem Schlüsselwort `ABORT` an den Anfang Ihres Skripts stellen:

```
$ chat -v ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ...
```

Auf ähnliche Weise können Sie den Timeout-Wert für Teile des Chat-Skripts ändern, indem Sie eine entsprechende *TIMEOUT*-Optionen einfügen.

Manchmal sollen auch Teile des Skripts nur ausgeführt werden, wenn eine bestimmte Bedingung erfüllt ist. Wenn Sie zum Beispiel den Login-Prompt vom anderen Ende nicht empfangen können, wollen Sie möglicherweise ein BREAK oder einen Wagenrücklauf übertragen. Sie erreichen dies, indem Sie ein UnterSkript an einen Expect-String anhängen. Dieses besteht genau wie beim normalen Skript auch aus einer

Reihe von Send- und Expect-Strings, die durch Bindestriche voneinander getrennt sind. Das UnterSkript wird immer dann ausgeführt, wenn der erwartete String, dem es anhängt, nicht innerhalb der vereinbarten Zeit eintrifft. Im obigen Beispiel würden Sie das Skript wie folgt modifizieren:

```
ogin:-BREAK-ogin: ppp ssword: GaGariN
```

Empfängt *chat* nun vom entfernten System nicht den Login-Prompt, wird das UnterSkript ausgeführt. Dabei wird zuerst ein BREAK übertragen, und danach wird erneut auf den Login-Prompt gewartet. Erscheint der Prompt nun, läuft das Skript weiter, als wäre nichts gewesen, ansonsten wird es mit einem Fehler abgebrochen.

Debuggen Ihres PPP-Setups

Per Voreinstellung schickt *pppd* alle Warnungen und Fehlermeldungen an die `daemon`-Einrichtung des *syslog*-Dämons. Sie müssen eine Zeile zu Ihrer *syslog.conf* hinzufügen, damit diese die Nachrichten in eine Datei oder sogar direkt auf die Konsole schreibt. Anderenfalls sortiert *syslog* sie einfach aus. Mit dem folgenden Eintrag werden alle Meldungen in die Datei */var/log/ppp-log* geschrieben:

```
daemon.*                    /var/log/ppp-log
```

Wenn Ihr PPP-Setup nicht auf Anhieb läuft, sollte Ihnen ein Blick in die Logdatei einen ersten Hinweis auf den möglichen Fehler liefern. Falls das nicht hilft, können Sie mit der Option *debug* eine gesonderte Debug-Ausgabe einschalten. Das veranlaßt *pppd*, den Inhalt aller empfangenen und gesendeten Kontrollpakete *syslog* aufzuzeichnen.

Wenn auch das nicht weiterhilft, bleibt noch die drastischste Möglichkeit, das Debugging auf Kernel-Ebene zu aktivieren, indem Sie *pppd* mit der Option *kdebug* ausführen. Die Option benötigt ein zusätzliches numerisches Argument, das eine bitweise ODER-Verknüpfung der folgenden Werte sein kann: 1 für allgemeine Debug-Nachrichten, 2 für die Ausgabe der Inhalte aller eingehenden HDLC-Frames und 4 für die Ausgabe aller ausgehenden HDLC-Frames. Um Kernel-Debugging-Nachrichten abfangen zu können, brauchen Sie entweder einen *syslogd*-Dämon, der die Datei */proc/kmsg* liest oder den *klogd*-Dämon. Beide leiten das Kernel-Debugging an die `kernel`-Einrichtung von *syslog* weiter.

IP-Konfigurations-Optionen

IPCP wird verwendet, um eine Reihe von IP-Parametern während der (Leitungs-) Konfigurationsphase zu vereinbaren. Üblicherweise sendet jede Seite ein Paket mit einer IPCP-Konfigurations-Anforderung, die beschreibt, welche Werte von den Defaults abweichen und auf welchen Wert sie gesetzt werden sollen. Nach dem Empfang prüft jede Seite die entsprechenden Optionen und stimmt entweder zu oder weist sie zurück.

pppd gibt Ihnen eine sehr große Kontrolle darüber, welche IPCP-Optionen es auszuhandeln versucht und welche nicht. Sie können dies über verschiedene Kommandozeilen-Optionen einstellen, die wir nachfolgend vorstellen wollen.

Wahl von IP-Adressen

Im obigen Beispiel haben wir *pppd* das System **c3po** anwählen lassen und einen IP-Link eingerichtet. Dabei wurden keinerlei Vorkehrungen getroffen, welche IP-Adressen an beiden Enden verwendet werden sollten.

Statt dessen haben wir die Adresse von **vlager** als lokale IP-Adresse verwendet und **c3po** sich eine eigene auswählen lassen. Trotzdem kann es manchmal sinnvoll sein, die Kontrolle darüber zu haben, welche Adresse am jeweiligen Ende verwendet wird. *pppd* bietet zu diesem Zweck eine Reihe von Optionen an.

Um bestimmte Adressen anzufordern, müssen Sie *pppd* diese mit der folgenden Option bekanntgeben:

`local_addr:remote_addr`

`local_addr` und `remote_addr` können dabei entweder in Dotted Quad Notation oder als Hostnamen angegeben werden.(7) Bei dieser Option geht *pppd* davon aus, daß es sich bei der ersten um seine eigene und bei der zweiten um die Adresse der anderen Seite handelt. Lehnt eine Seite eine der Adressen während der IPCP-Vereinbarungsphase ab, wird kein IP-Link aufgebaut.(8)

Wenn Sie nur die lokale Adresse setzen möchten, gleichzeitig aber die Adresse akzeptieren wollen, die die Gegenseite verwendet, lassen Sie den `remote_addr`-Teil einfach weg. Soll **vlager** beispielsweise die IP-Adresse **130.83.4.27** anstelle seiner eigenen verwenden, können Sie *130.83.4.27:* in der Kommandozeile eingeben. Genauso gehen Sie vor, wenn Sie nur die Adresse der anderen Seite einstellen wollen, indem Sie das Feld `local_addr` leer lassen. Per Standardeinstellung verwendet *pppd* dann die Adresse, die mit Ihrem Hostnamen verknüpft ist.

Einige PPP-Server, die viele Client-Sites verwalten müssen, weisen Adressen dynamisch zu. Adressen werden an Systeme nur vergeben, wenn ein Anruf erfolgt, und werden anschließend recyclet, sobald die Verbindung wieder unterbrochen wird. Wenn Sie einen solchen Server anwählen, müssen Sie darauf achten, daß Sie keine bestimmte Adresse vom Server anfordern, sondern die Ihnen zugedachte akzeptieren. Das bedeutet, daß Sie das Argument `local_addr` nicht verwenden dürfen. Gleichzeitig müssen Sie auch die Option *noipdefault* verwenden, mit der Sie *pppd* anweisen, auf eine IP-Adresse von der Gegenseite zu warten, statt die lokale Host-Adresse zu verwenden.

Routen über einen PPP-Link

Nachdem das Netzwerk-Interface eingerichtet ist, setzt *pppd* nur eine Host-Route zur Gegenseite. Wenn sich der entfernte Host in einem LAN befindet, wollen Sie möglicherweise auch in der Lage sein, mit Hosts »hinter« diesem Punkt zu kommunizieren. Darum muß eine Netzwerkroute eingerichtet werden.

Sie haben bereits gesehen, daß Sie *pppd* mit der Option *defaultroute* anweisen können, die Default-Route einzurichten. Diese Option ist sehr nützlich, wenn der von Ihnen angewählte PPP-Server als Internet-Gateway fungiert.

Der entgegengesetzte Fall, bei dem Ihr System als Gateway für einen einzelnen Host fungiert, ist relativ einfach zu erreichen. Nehmen wir zum Beispiel einen Angestellten der virtuellen Brauerei, dessen zu Hause stehende Maschine **loner** heißt. Wird über PPP eine Verbindung mit **vlager** aufgebaut, verwendet er eine Adresse des Brauerei-Subnetzes. Auf **vlager** können wir *pppd* nun mit der Option *proxyarp* aufrufen, was einen sog. Proxy-ARP-Eintrag für **loner** installiert (Proxy bedeutet soviel wie Vollmacht). Auf diese Weise wird **loner** automatisch für alle Hosts in der Brauerei und Winzerei zugänglich gemacht.

Leider sind die Dinge nicht immer so einfach. Beispielsweise wird bei der Verbindung von zwei lokalen Netzwerken eine bestimmte Netzwerkroute benötigt, weil beide Netzwerke ihre persönliche Default-Route haben könnten. Würden beide Seiten den PPP-Link als Default-Route verwenden, würde das außerdem eine

Schleife erzeugen, bei der Pakete mit unbekanntem Ziel solange hin- und hertransportiert würden, bis deren »Time-to-live« überschritten wäre.

Nehmen wir als Beispiel einmal an, daß die virtuelle Brauerei eine Zweigstelle in einer anderen Stadt eröffnet. Dort wird ein eigenes Ethernet mit der IP-Netzwerknummer **172.16.3.0** betrieben, das als Subnetz 3 im Klasse-B-Netzwerk der Brauerei geführt wird. Die Filiale möchte sich an das Hauptnetz der Brauerei über PPP anschließen, um Kundendaten zu aktualisieren etc. Erneut dient **vlager** als Gateway. Die Gegenseite verwendet dabei den Namen **sub-etha** und die IP-Adresse **172.16.3.1.**.

Sobald **sub-etha** die Verbindung zu **vlager** herstellt, legt es wie gewöhnlich eine Default-Route an, die auf **vlager** zeigt. Gleichzeitig müssen wir aber auf **vlager** eine Netzwerkroute für Subnetz 3 einrichten, die über **sub-etha** läuft. Dazu verwenden wir ein *pppd*-Feature, das wir bisher noch nicht besprochen haben -- den *ip-up*-Befehl. Dabei handelt es sich um ein Shell-Skript oder Programm im Verzeichnis */etc /ppp*, das ausgeführt wird, nachdem das PPP-Interface konfiguriert wurde. Wenn vorhanden, wird es mit den folgenden Parametern aufgerufen:

```
ip-up iface device speed local_addr remote_addr
```

iface steht für das zu verwendende Netzwerk-Interface, `device` ist der Pfadname auf die verwendete serielle Gerätedatei (*/dev/tty*, wenn stdin/stdout verwendet werden), und `speed` steht für die Geschwindigkeit der seriellen Schnittstelle. `local_addr` und `remote_addr` enthalten die IP-Adressen beider Enden des Links in Dotted Quad Notation. In unserem Fall könnte das *ip-up*-Skript das folgende Kode-Fragment enthalten:

```
#!/bin/sh
case $5 in
172.16.3.1)            # sub-etha
        route add -net 172.16.3.0 gw 172.16.3.1;;
...
esac
exit 0
```

Auf dieselbe Weise wird */etc/ppp/ip-down* verwendet, um alle Aktionen von *ip-up* wieder rückgängig zu machen, nachdem der PPP-Link wieder unterbrochen wurde.

Nun, das Routing-Schema ist noch nicht vollständig. Wir haben zwar Einträge in die Routing-Tabellen der beiden Hosts aufgenommen, aber bis jetzt wissen die Hosts in beiden Netzwerken noch nichts über den PPP-Link. Das ist kein Problem, wenn alle Hosts in der Zweigstelle die Default-Route auf **sub-etha** zeigen lassen und alle Hosts der Brauerei standardmäßig auf **vlager** routen. Wenn dies nicht der Fall ist, ist die einzige Möglichkeit üblicherweise die Verwendung eines Routing-Dämons wie *gated*. Nachdem die Netzwerkroute auf **vlager** eingerichtet ist, gibt der Routing-Dämon die neue Route an alle Hosts der angebundenen Subnetze weiter.

Link-Kontrolloptionen

LCP, das Link Control Protocol, haben wir ja bereits vorgestellt. Es wird genutzt, um die Charakteristika eines Links zu vereinbaren und den Link zu testen.

Die beiden wichtigsten von LCP zu vereinbarenden Optionen sind die »Maximum Receive Unit« und die »Asynchronous Control Character Map«. Es gibt noch eine ganze Reihe weiterer LCP-Konfigurationsoptionen, die aber bei weitem zu spezialisiert sind, als daß wir sie hier besprechen

könnten.

Die »Asynchronous Control Character Map«, allgemein *Async-Map* genannt, wird bei asynchronen Links wie beispielsweise Telefonleitungen verwendet, um Steuerzeichen zu erkennen, die durch eine Zwei-Zeichen-Sequenz (Escape-Sequenz) ersetzt werden müssen. Beispielsweise könnten Sie die beim Software-Handshake verwendeten Zeichen XON und XOFF umgehen wollen, weil einige fehlerhaft konfigurierte Modems sich beim Empfang von XOFF verabschieden. Ein weiterer Kandidat ist Ctrl-] (das *telnet*-Escape-Zeichen). PPP erlaubt es Ihnen, solche Zeichen mit den ASCII-Kodes 0 bis 31 besonders zu kennzeichnen, indem Sie sie in die Async-Map aufnehmen.

Die Async-Map ist eine 32 Bit breite Bitmap, bei der das niederwertigste Bit dem ASCII-Zeichen NUL und das höherwertigste Bit dem ASCII-Wert 31 entspricht. Ist ein Bit gesetzt, bedeutet das, daß das entsprechende Zeichen erst umgewandelt werden muß, bevor es über den Link transportiert werden kann. Am Anfang steht die Async-Map auf *0xffffffff*, alle Steuerzeichen müssen umgewandelt werden.

Um Ihrer Seite mitzuteilen, daß nicht alle Steuerzeichen umgewandelt werden müssen, können Sie *pppd* mit der Option *asyncmap* eine neue Async-Map übergeben. Wenn beispielsweise nur die Steuerzeichen `^S` und `^Q` (ASCII 17 und 19 werden häufig als XON und XOFF verwendet) umgewandelt werden müssen, können Sie die folgende Option nutzen:

```
asyncmap 0x000A0000
```

Mit der »Maximum Receive Unit«, oder MRU, signalisieren wir die maximale Größe eines HDLC-Frames, den wir zu empfangen bereit sind. Obwohl Sie das an den MTU-Wert (Maximum Transfer Unit) erinnern könnte, haben diese beiden doch sehr wenig gemeinsam. Die MTU ist ein Parameter der Kernel-Netzwerkeinheit und beschreibt die maximale Größe, die das Interface verarbeiten kann. Die MRU ist eher eine Empfehlung an die andere Seite, keine Frames zu generieren, die länger als die MRU sind. Das Interface muß aber dennoch in der Lage sein, Frames mit einer Größe von bis zu 1500 Byte zu empfangen.

Die Wahl einer MRU ist daher also nicht so sehr eine Frage dessen, was ein Link zu übertragen in der Lage ist, sondern eher, wie der beste Durchsatz zu erzielen ist. Wenn Sie mit interaktiven Anwendungen arbeiten wollen, sollten Sie die MRU auf einen so niedrigen Wert wie 296 setzen, damit gelegentliche größere Pakete (sagen wir von einer FTP-Session) Ihren Cursor nicht zum »ruckeln« bringen. Sie weisen *pppd* an, eine MRU von 296 zu verwenden, indem Sie die Option *mru 296* in der Kommandozeile übergeben. Kleine MRUs machen allerdings nur Sinn, wenn Sie die VJ-Header-Komprimierung nicht deaktiviert haben (sie ist per Standardeinstellung aktiviert).

pppd versteht auch eine ganze Reihe von LCP-Optionen, mit denen Sie das gesamte Verhalten der Verhandlungsphase konfigurieren können, wozu beispielsweise die maximale Anzahl von Konfigurations-Anforderungen gehört, die ausgetauscht werden können, bevor die Verbindung beendet wird. Solange Sie nicht genau wissen, was Sie da tun, sollten Sie aber die Finger von diesen Optionen lassen.

Zum Schluß gibt es noch zwei Optionen, die sich mit LCP-Echo-Nachrichten beschäftigen. PPP definiert zwei besondere Nachrichten: »Echo Request« und »Echo Response«. *pppd* verwendet diese, um zu prüfen, ob der Link noch funktioniert. Sie aktivieren diese Option durch *lcp-echo-interval* und die Angabe einer Zeit in Sekunden. Werden vom entfernten Host innerhalb dieser Zeitspanne keine Frames empfangen, erzeugt *pppd* einen »Echo Request« und erwartet, daß die Gegenseite mit einer »Echo Response« antwortet. Erfolgt von der Gegenstelle keine Antwort, wird die Verbindung nach einer bestimmten Anzahl von Versuchen unterbrochen. Diese Anzahl können Sie mit der Option *lcp-echo-failure* bestimmen. Per Standardeinstellung ist dieses Feature deaktiviert.

Allgemeine Sicherheitsaspekte

Ein fehlerhaft konfigurierter PPP-Dämon kann eine vernichtende Sicherheitslücke sein. Im schlimmsten Fall kann das so sein, als ob sich jeder direkt an Ihr Ethernet anschließen könnte (und das ist sehr schlecht). In diesem Abschnitt wollen wir einige Maßnahmen diskutieren, die Ihre PPP-Konfiguration sicher machen sollten.

Ein Problem mit *pppd* ist, daß es zur Konfiguration des Netz-Interfaces und der Routing-Tabelle **root**-Privilegien benötigt. Üblicherweise lösen Sie dieses Problem, indem Sie es mit Setuid **root** ausführen. Jedoch erlaubt es *pppd* dem Benutzer, verschiedene sicherheitsrelevante Optionen zu setzen. Um sich vor Angriffen zu schützen, die ein Benutzer durch Manipulation dieser Optionen starten könnte, wird empfohlen, eine Reihe von Voreinstellungen in der globalen Datei */etc/ppp /options* einzutragen. Ein Beispiel dafür finden Sie im Abschnitt »*Arbeiten mit Optionsdateien*«. Einige dieser Optionen, wie die zur Authentisierung, können dann vom Benutzer nicht überschrieben werden, was eine vernünftige Sicherung gegen Manipulationen darstellt.

Natürlich müssen Sie sich aber auch selbst vor dem System schützen, mit dem Sie über PPP kommunizieren. Um zu verhindern, daß sich Hosts als jemand anders ausgeben, sollten Sie immer irgendeine Form der Authentisierung von der Gegenseite verlangen. Darüber hinaus sollten Sie es fremden Hosts nicht erlauben, eine beliebige von ihnen gewählte IP-Adresse zu verwenden, sondern sie auf ein paar wenige einschränken. Der folgende Abschnitt befaßt sich mit diesen Themen.

Authentizierung mit PPP

CHAP verglichen mit PAP

Mit PPP kann jedes System seine Gegenseite auffordern, sich zu authentisieren. Dazu kann eins von zwei Authentizierungs-Protokollen verwendet werden. Dies ist zum einen das »Password Authentication Protocol« (PAP) und zum anderen das »Challenge Handshake Authentication Protocol« (CHAP). Sobald eine Verbindung steht, kann jede Seite die andere auffordern, sich zu authentisieren, gleichgültig, ob sie die rufende oder angerufene Seite ist. Im folgenden werde ich von »Client« und »Server« reden, um zwischen dem sich authentisierenden System und dem Authentikator zu unterscheiden. Ein PPP-Dämon kann von seinem Gegenüber die Authentisierung anfordern, indem er einfach eine weitere LCP-Konfigurations-Anforderung sendet, die das verwendete Authentisierungs-Protokoll enthält.

PAP arbeitet grundsätzlich auf dieselbe Weise wie die normale Login-Prozedur. Der Client authentiziert sich, indem er einen Benutzernamen und ein (optional verschlüsseltes) Paßwort an den Server schickt, die der Server dann beide mit seiner sog. Secrets-Datenbank vergleicht. Diese Technik ist durch Lauscher angreifbar, die an der seriellen Leitung horchen. Auch Angriffe nach dem Trial-and-Error-Verfahren sind hier denkbar.

CHAP hat diese Defizite nicht. Bei CHAP sendet der Authentikator (d. h. der Server) einen zufällig erzeugten »Anforderungs-String« zusammen mit seinem Hostnamen an den Client. Der Client verwendet den Hostnamen, um das entsprechende »Secret« nachzusehen, kombiniert es mit der Anforderung und verschlüsselt den String mit Hilfe einer Einweg-Hash-Funktion. Das Ergebnis wird zusammen mit dem Hostnamen des Client an den Server zurückgeschickt. Der Server führt nun dieselbe Berechnung durch und akzeptiert den Client, wenn er zu demselben Ergebnis kommt.

Eine weitere Eigenschaft von CHAP besteht darin, daß es die Authentizierung durch den Client nicht nur beim Start abfragt, sondern solche Anforderungen in regelmäßigen Zeitabständen sendet, um sicherzustellen, daß der Client nicht durch einen Eindringling ersetzt wurde, der beispielsweise nur die Telefonleitung umgeschaltet hat.

pppd speichert die geheimen Schlüssel für CHAP und PAP in zwei separaten Dateien namens */etc/ppp/chap-secrets* und *pap-secrets*. Durch Eintragen eines entfernten Host in die eine oder andere Datei haben Sie die Kontrolle darüber, ob CHAP oder PAP bei der Authentisierung verwendet wird.

pppd ist so voreingestellt, daß es von der Gegenseite keine Authentizierung benötigt; fordert die Gegenstelle eine Authentizierung an, wird diese aber durchgeführt. Weil CHAP soviel strenger ist als PAP, versucht *pppd*, wann immer möglich, auf das erste Verfahren zurückzugreifen. Wird dies von der Gegenseite nicht unterstützt, oder kann *pppd* kein CHAP-Secret für den anderen Rechner in seiner *chap-secrets* finden, wechselt es zu PAP. Kann auch kein PAP-Secret für die andere Seite gefunden werden, wird die Authentizierung vollständig abgebrochen. Die Konsequenz daraus ist, daß auch die Verbindung unterbrochen wird.

Dieses Verhalten kann auf verschiedene Weise angepaßt werden. Wird beispielsweise die Option *auth* verwendet, verlangt *pppd* von der Gegenseite eine Authentizierung. *pppd* kann dabei entweder auf CHAP oder auf PAP zurückgreifen, solange das Secret der anderen Seite in der CHAP- oder PAP-Datenbank gefunden wird. Es gibt andere Optionen, mit denen auf ein bestimmtes Authentizierungs-Protokoll zurückgegriffen wird. Ich gehe an dieser Stelle aber nicht auf sie ein.

Wenn alle Systeme, mit denen Sie sich über PPP unterhalten, bereit sind, sich Ihnen gegenüber zu authentisieren, sollten Sie die Option *auth* in die globale Datei */etc /ppp/options* aufnehmen und Paßwörter für alle Systeme in die Datei *chap-secrets* eintragen. Wird CHAP von einem System nicht unterstützt, fügen Sie den Eintrag in die Datei *pap-secrets* ein. Auf diese Weise stellen Sie sicher, daß keine unautorisierten Systeme mit Ihrem Host anbandeln.

Die nächsten beiden Abschnitte diskutieren die beiden PPP-Secrets-Dateien *pap-secrets* und *chap-secrets*. Diese Dateien sind im Verzeichnis */etc/ppp* zu finden und enthalten Dreiergruppen aus Clients, Servern und Paßwörtern, denen optional eine Liste mit IP-Adressen folgt. Die Interpretation der Client- und Server-Felder ist bei CHAP und PAP unterschiedlich und hängt auch davon ab, ob wir uns bei der Gegenseite authentizieren, oder ob wir den Server auffordern, sich bei uns zu authentizieren.

Die CHAP-Secrets-Datei

Wenn *pppd* sich über CHAP einem Server gegenüber ausweisen muß, durchsucht es die Datei *chap-secrets* nach einem Eintrag, bei dem das Client-Feld mit dem lokalen Hostnamen und das Server-Feld mit dem Hostnamen des Servers (während der CHAP-Anforderung mit übertragen) übereinstimmen. Muß sich die Gegenseite authentizieren, sind die Regeln einfach umgekehrt: *pppd* sucht dann nach einem Eintrag, bei dem das Client-Feld mit dem Hostnamen des entfernten Rechners (übertragen während der CHAP-Antwort des Client), und das Server-Feld mit dem lokalen Hostnamen übereinstimmen.

Nachfolgend ein einfaches Beispiel einer *chap-secrets*-Datei für **vlager**:(9)

```
# CHAP-Secrets für vlager.vbrew.com
#
# Client         Server         Secret               Adressen
```

```
#----------------------------------------------------------------------
vlager.vbrew.com  c3po.lucas.com    "Use The Source Luke" vlager.vbrew.com
c3po.lucas.com    vlager.vbrew.com  "riverrun, pasteve"   c3po.lucas.com
*                 vlager.vbrew.com  "VeryStupidPassword"  pub.vbrew.com
```

Wird eine PPP-Verbindung zu **c3po** aufgebaut, fordert **c3po vlager** auf, sich zu authentizieren, indem es eine CHAP-Anforderung überträgt. *pppd* durchsucht dann *chap-secrets* nach einem Eintrag, bei dem das Client-Feld **vlager.vbrew.com** und das Server-Feld **c3po.lucas.com** enthalten,(10) und findet dabei die erste oben aufgeführte Zeile. Daraufhin wird eine CHAP-Antwort aus dem Anforderungs-String und dem Secret (`Use The Source Luke`) erzeugt und an **c3po** geschickt.

Zur selben Zeit baut *pppd* eine CHAP-Anforderung für **c3po** auf, die einen eindeutigen Anforderungs-String und den voll qualifizierten Hostnamen **vlager.vbrew.com** enthält. **c3po** erzeugt die entsprechende CHAP-Antwort auf die oben beschriebene Art und Weise und gibt diese an **vlager** zurück. *pppd* filtert sich nun den Hostnamen des Client (**c3po.vbrew.com**) aus der Antwort heraus und durchsucht die Datei *chap-secrets* nach einem Eintrag, bei dem **c3po** als Client und **vlager** als Server vorhanden sind. Das ist im zweiten Eintrag unserer Beispieldatei der Fall und *pppd* kombiniert die CHAP-Anforderung und das Secret `riverrun, pasteve`, verschlüsselt sie und vergleicht das Ergebnis mit der CHAP-Antwort von **c3po**.

Das vierte Feld ist optional und enthält eine Liste der IP-Adressen, die für den im ersten Feld aufgeführten Client akzeptabel sind. Die Adressen können in Dotted Quad Notation oder als Hostnamen angegeben werden, die mit Hilfe des Resolvers aufgelöst werden. Fordert beispielsweise **c3po** eine bestimmte Adresse während der IPCP-Vereinbarung an, die nicht in dieser Liste steht, wird diese Anforderung abgelehnt und IPCP wird beendet. Bei unserer oben aufgeführten Beispieldatei ist **c3po** also auf die Verwendung seiner eigenen IP-Adresse eingeschränkt. Ist das Adreßfeld leer, ist jede Adresse erlaubt. Wird als Wert »`-`« eingetragen, wird IP für diesen Client gar nicht erst initialisiert.

Die dritte Zeile in unserer *chap-secrets* erlaubt es jedem Host, einen PPP-Link mit **vlager** aufzubauen, weil das Sternchen (`*`) als Platzhalter für jeden beliebigen Hostnamen steht. Die einzige Voraussetzung ist, daß er das Secret kennt und die Adresse von **pub.vbrew.com** verwendet. Einträge mit sochen Platzhaltern (Wildcards) können an jeder beliebigen Stelle in der Secrets-Datei stehen, weil *pppd* immer den Eintrag verwendet, der am ehesten zu einem Server/Client-Paar paßt.

pppd benötigt möglicherweise bei der Bildung von Hostnamen etwas Hilfe. Wie bereits vorher erläutert, ist der Name des anderen Host immer im CHAP-Anforderungs- oder Antwortpaket enthalten. Der lokale Hostname wird standardmäßig durch den Aufruf der Funktion *gethostname(2)* ermittelt. Wenn Sie den Systemnamen auf Ihren unqualifizierten Hostnamen gesetzt haben, müssen Sie *pppd* mit der Option *domain* zusätzlich noch den Domainnamen bekanntgeben:

```
# pppd … domain vbrew.com
```

Das hängt den Domainnamen der Brauerei bei allen authentizierungsbezogenen Aktivitäten an **vlager** an. Andere Optionen, mit denen Sie den lokalen Hostnamen für *pppd* anpassen können, sind *usehostname* und *name*. Wenn Sie die lokale IP-Adresse mit Hilfe von *`local:remote`* in der Kommandozeile angeben, und *`local`* einen Namen anstelle einer Dotted Quad enthält, nutzt *pppd* diesen als lokalen Hostnamen.

Die PAP-Secrets-Datei

Die Secrets-Datei für PAP ist der von CHAP sehr ähnlich. Die ersten beiden Felder enthalten einen Benutzer- und einen Servernamen. Das dritte Feld enthält das PAP-Secret. Fordert die Gegenseite die Authentizierung

an, verwendet *pppd* den Eintrag, bei dem das Server-Feld mit dem lokalen Hostnamen und das Benutzerfeld mit dem in der Anforderung übertragenen Benutzernamen übereinstimmen. Wenn es sich selbst authentizieren muß, wählt sich *pppd* das Secret aus dem Eintrag aus, bei dem das erste Feld den lokalen Hostnamen und das Server-Feld den entfernten Hostnamen enthält.

Eine PAP-Secrets-Datei könnte beispielsweise wie folgt aussehen:

```
# /etc/ppp/pap-secrets
#
# Benutzer      Server          Secret          Adressen
vlager-pap      c3po            cresspahl       vlager.vbrew.com
c3po            vlager          DonaldGNUth     c3po.lucas.com
```

Die erste Zeile wird von uns zur Authentizierung verwendet, wenn wir mit **c3po** kommunizieren. Die zweite Zeile beschreibt, wie ein Benutzer namens **c3po** sich uns gegenüber zu authentizieren hat.

Der Name `vlager-pap` in der ersten Spalte ist der Benutzername, den wir an **c3po** senden. Per Standardeinstellung wählt *pppd* den lokalen Hostnamen als Benutzernamen, Sie können aber auch einen anderen Namen bestimmen, indem Sie ihn mit der Option *user* übergeben.

Wenn ein Eintrag aus der Datei *pap-secrets* herausgesucht werden soll, um die Authentizierung mit der Gegenseite durchzuführen, muß *pppd* den Namen des anderen Host wissen. Weil es keine Möglichkeit hat, ihn herauszufinden, müssen Sie ihn mit Hilfe der Option *remotename* in der Kommandozeile mit angeben. Soll beispielsweise der obige Eintrag für die Authentizierung mit **c3po** verwendet werden, muß die *pppd*-Kommandozeile wie folgt aussehen:

```
# pppd ... remotename c3po user vlager-pap
```

Im vierten Feld (und in allen folgenden Feldern) können Sie, genau wie bei der CHAP-Secrets-Datei auch, die IP-Adressen angeben, die für einen bestimmten Host erlaubt sind. Die Gegenstelle darf dann nur Adressen aus dieser Liste anfordern. In unserer Beispieldatei verlangen wir, daß **c3po** seine echte IP-Adresse benutzt.

Denken Sie daran, daß PAP eine eher schwache Authentizierungs-Methode ist, und daß Sie, wenn möglich, immer auf CHAP zurückgreifen sollten. Aus diesem Grund gehen wir hier nicht detaillierter auf PAP ein. Wenn Sie mehr über PAP erfahren wollen, finden Sie weitere Informationen in der *pppd(8)*-Manpage.

Konfiguration eines PPP-Servers

Wenn Sie *pppd* als Server ausführen wollen, müssen Sie nur die entsprechenden Optionen in der Kommandozeile angeben. Stellen Sie sich vor, Sie generieren einen speziellen Account, z. B. ppp, dem Sie dann ein Skript oder ein Programm als Login-Shell zuweisen, das *pppd* mit den entsprechenden Optionen startet. Beispielsweise könnten Sie die folgende Zeile in die */etc/passwd* einfügen:

```
ppp:*:500:200:Public PPP Account:/tmp:/etc/ppp/ppplogin
```

Natürlich können Sie andere UIDs und GIDs verwenden, als oben aufgeführt. Außerdem müssen Sie mit dem Befehl *passwd* noch das entsprechende Paßwort für diesen Account zuweisen.

Das Skript *ppplogin* könnte dann wie folgt aussehen:

```
#!/bin/sh
# ppplogin -- Skript zum automatischen Starten von pppd beim Login
mesg n
stty -echo
exec pppd -detach silent modem crtscts
```

Der Befehl *mesg* verhindert, daß andere Benutzer an den tty schreiben können (beispielsweise mit *write*). Der Befehl *stty* schaltet das Echo aus. Das ist notwendig, weil sonst alle von der Gegenseite übertragenen Daten nochmal zurückgeschickt würden. Die wichtigste der oben angegebenen *pppd*-Optionen ist *-detach*, weil sie verhindert, daß sich *pppd* vom kontrollierenden tty abtrennt. Wenn Sie diese Option nicht angeben, würde es in den Hintergrund treten und das Shell-Skript beenden. Das würde wiederum dazu führen, daß die serielle Leitung abgebaut würde, und somit die Verbindung beendet wäre. Mit der *silent*-Option weisen Sie *pppd* an zu warten, bis ein Paket vom anrufenden System empfangen wurde, bevor Sie mit dem Senden beginnen. Auf diese Weise werden Übertragungs-Timeouts verhindert, die auftreten können, wenn das anrufende System seinen PPP-Client nur langsam hochfährt. Mit der Modem-Option übernimmt *pppd* die Kontrolle der Modem-Steuerungsleitungen auf dem seriellen Port. Diese Option sollten Sie immer aktivieren, wenn Sie *pppd* mit einem Modem betreiben. Die Option *crtscts* schaltet den Hardware-Handshake ein.

Neben diesen Optionen sollten Sie auch irgendeine Form der Authentizierung aktivieren. Dazu können Sie die Option *auth* in der *pppd*-Kommandozeile oder in den globalen Optionsdateien setzen. Die Manpage beschreibt auch spezifischere Optionen, mit denen Sie bestimmte Authentizierungs-Protokolle ein- und ausschalten können.

Fußnoten

(1)

Die relevanten RFCs sind in der Bibliographie am Ende dieses Buches aufgeführt.

(2)

Tatsächlich ist HDLC ein wesentlich allgemeineres Protokoll, das sich die International Standards Organization (ISO) ausgedacht hat.

(3)

Beide Autoren haben angedeutet, daß sie in naher Zukunft sehr beschäftigt sein werden. Wenn Sie irgendwelche Fragen zu PPP im allgemeinen haben, sollten Sie besser die Leute auf der linux-ppp-Mailing-Liste fragen.

(4)

karl@morningstar.com

(5)

Die Default-Netzwerkroute wird nur installiert, wenn noch keine vorhanden ist.

(6)

Wenn Sie syslog.conf so editieren, daß der Log in einer Datei gespeichert wird, müssen Sie darauf achten, daß diese Datei nicht von jedem gelesen werden kann, weil chat per Voreinstellung auch das gesamte Skript aufzeichnet einschließlich aller Paßwörter etc.

(7)

Die Verwendung von Hostnamen bei dieser Option hat Auswirkungen für die CHAP-Authentisierung. Mehr dazu finden Sie im Abschnitt zu CHAP in diesem Kapitel.

(8)

Sie können der anderen PPP-Seite erlauben, Ihre Vorstellung einer IP-Adresse zu überschreiben, indem Sie pppd mit den Optionen ipcp-accept-local und ipcp-accept-remote ausführen.

(9)

Die Anführungszeichen sind nicht Teil des Paßworts, sondern dienen nur dazu, die Leerzeichen innerhalb des Paßworts zu schützen.

(10)

Der Hostname stammt aus der CHAP-Anforderung.

Kapitel 9
Wichtige NetzwerkFeatures

Nachdem Sie IP und den Resolver erfolgreich eingerichtet haben, müssen Sie sich den Diensten zuwenden, die Sie in Ihrem Netzwerk anbieten wollen. Dieses Kapitel behandelt die Konfiguration einiger einfacher Netzwerk-Anwendungen, einschließlich des *inetd*-Servers und der Programme der *rlogin*-Familie. Das »Remote Procedure Call«-Interface, auf dem Dienste wie das Network File System (NFS) und das Network Information System (NIS) basieren, wird auch kurz behandelt. Die Konfiguration von NFS und NIS nimmt aber wesentlich mehr Raum in Anspruch und wird daher in separaten Kapiteln besprochen. Das gleiche gilt auch für Elektronische Post und Netnews.

Leider können wir in diesem Buch nicht alle Netzwerk-Anwendungen besprechen. Wenn Sie eine der hier nicht behandelten Anwendungen wie *talk*, *gopher* oder *Mosaic* installieren wollen, müssen wir Sie auf die entsprechenden Manpages verweisen.

Der inetd Super-Server

Häufig werden Dienste durch sogenannte *Dämonen* bereitgestellt. Ein Dämon ist ein Programm, das einen bestimmten Port öffnet und auf eingehende Verbindungen wartet. Wird eine solche Verbindung aufgebaut, erzeugt der Dämon einen Kind-Prozeß, der die Verbindung übernimmt, während der Parent selbst weiter nach eingehenden Anforderungen Ausschau hält. Dieses Konzept hat den Nachteil, daß für jeden angebotenen Service ein entsprechender Dämon laufen muß, der am Port nach Anforderungen horcht, was grundsätzlich eine Verschwendung von Systemressourcen nach sich zieht, wie beispielsweise den Verbrauch von Swap-Speicher.

Daher verwenden nahezu alle UNIX-Installationen einen »Super-Server«, der Sockets für eine Reihe von Diensten erzeugt und mit Hilfe des *select(2)*-Systemaufrufs simultan abhört. Fordert ein entfernter Host einen Dienst an, wird dies vom Super-Server registriert, und er startet den für diesen Port zuständigen Server.

Der im allgemeinen verwendete Super-Server ist *inetd*, der »Internet Dämon«. Er wird während der Bootphase des Systems gestartet und liest eine Liste der Dienste, die er verwalten soll, aus der Datei */etc/inetd.conf*. Zusätzlich zu den von *inetd* gesteuerten Servern gibt es eine Reihe trivialer Dienste, sogenannte *interne Dienste*, die *inetd* selbständig ausführt. Dazu gehören beispielsweise *chargen*, das einfach eine Zeichenkette erzeugt, und *daytime*, das die nach Meinung des Systems aktuelle Tageszeit zurückliefert.

Ein Eintrag in dieser Datei besteht aus einer einzelnen Zeile, die sich aus den folgenden Feldern zusammensetzt:

```
service type protocol wait user server cmdline
```

Die Bedeutung der einzelnen Felder wird nachfolgend erklärt:

service

Enthält den Namen des Dienstes. Dieser Name muß in eine Portnummer übersetzt werden, was durch einen Lookup in der Datei */etc/services* geschieht. Diese Datei wird im Abschnitt »*Die Dateien services und protocols*« beschrieben.

type

Bestimmt den Socket-Typ entweder als `stream` (bei verbindungsorientierten Protokollen) oder als `dgram` (für Datagramm-Protokolle). TCP-basierte Dienste sollten daher immer `stream` verwenden, während UDP-basierte Dienste immer `dgram` verwenden sollten.

protocol

Benennt das vom Dienst verwendete Transportprotokoll. Hier muß ein gültiger, in der (weiter unten erläuterten) Datei *protocols* enthaltener Name stehen.

wait

Diese Option gilt nur für `dgram`-Sockets gültig. Sie kann entweder `wait` oder `nowait` lauten. Wird `wait` angegeben, führt inetd nur jeweils einen Server für den angegebenen Port aus. Anderenfalls beginnt es wieder umgehend an diesem Port zu horchen, sobald der Server gestartet wurde.

Das ist sinnvoll bei sog. »Single-Threaded-Servern«, die alle eingehenden Datagramme lesen, bis keine weiteren mehr ankommen, und sich dann automatisch beenden. Die meisten RPC-Server sind von dieser Art und sollten daher immer `wait` verwenden. Der entgegengesetzte Typ, der »Multi-Threaded-Server«, erlaubt eine unbeschränkte Anzahl von gleichzeitig laufenden Instanzen und wird nur selten benutzt. Dieser Server-Typ sollte `nowait` angeben.

`stream`-Sockets sollten immer `nowait` verwenden.

user

Hier steht die Login-ID des Benutzers, unter der der Prozeß ausgeführt wird. Das wird häufig root sein, einige Dienste verwenden aber auch andere Accounts. Sie sollten übrigens immer das Prinzip der geringsten Privilegien anwenden, d.h, ein Befehl sollte nur mit Privilegien ausgestattet sein, die er zur fehlerfreien Ausführung unbedingt benötigt. Zum Beispiel läuft der NNTP-News-Server als news, während Dienste, die ein Sicherheitsrisiko darstellen (z. B. *tftp* oder *finger*), häufig als nobody laufen.

server

Enthält den vollen Pfadnamen auf das auszuführende Server-Programm. Interne Dienste werden mit dem Schlüsselwort `internal` gekennzeichnet.

cmdline

Die an den Server zu übergebende Befehlszeile. Diese enthält auch das Argument 0, den Befehlsnamen. Üblicherweise ist dies der Programmname des Servers, es sei denn, der Server verhält sich anders, wenn er unter einem anderen Namen gestartet wird.

Dieses Feld ist bei internen Diensten leer.

Ein Beispiel für *inetd.conf* ist in Beispiel 9--1 zu sehen. Der *finger*-Service ist auskommentiert, so daß er nicht zur Verfügung steht. Dies wird häufig aus Sicherheitsgründen getan, weil er von Eindringlingen dazu verwendet werden kann, Namen der Benutzer Ihres Systems zu ermitteln. *Beispiel 9-1. /etc/inetd.conf Beispieldatei*

```
#
# inetd-Dienste
ftp       stream tcp nowait root   /usr/sbin/ftpd    in.ftpd -l
telnet    stream tcp nowait root   /usr/sbin/telnetd in.telnetd -b/etc/issue
#finger    stream tcp nowait bin    /usr/sbin/fingerd in.fingerd
#tftp      dgram  udp wait   nobody /usr/sbin/tftpd   in.tftpd
```

```
#tftp      dgram  udp wait   nobody /usr/sbin/tftpd   in.tftpd /boot/diskless
login     stream tcp nowait root   /usr/sbin/rlogind in.rlogind
shell     stream tcp nowait root   /usr/sbin/rshd    in.rshd
exec      stream tcp nowait root   /usr/sbin/rexecd  in.rexecd
#
#       inetd-interne Dienste
#
daytime   stream tcp nowait root internal
daytime   dgram  udp nowait root internal
time      stream tcp nowait root internal
time      dgram  udp nowait root internal
echo      stream tcp nowait root internal
echo      dgram  udp nowait root internal
discard   stream tcp nowait root internal
discard   dgram  udp nowait root internal
chargen   stream tcp nowait root internal
chargen   dgram  udp nowait root internal
```

tftp ist ebenfalls auskommentiert. *tftp* implementiert das *Trivial File Transfer Protocol* (TFTP), das es einem erlaubt, allgemein lesbare Dateien ohne Paßwortprüfung etc. von Ihrem System zu übertragen. Das ist besonders bei der Datei */etc/passwd* gefährlich, um so mehr, wenn Sie nicht mit Shadow-Paßwörtern arbeiten.

TFTP wird üblicherweise von Clients ohne eigenes Diskettenlaufwerk und von X-Terminals genutzt, um deren Kode vom einem Bootserver herunterzuladen. Muß *tftpd* aus diesem Grund ausgeführt werden, sollten Sie sicherstellen, daß Sie den Zugriff nur auf solche Verzeichnisse erlauben, aus denen die Clients die entsprechenden Dateien lesen. Diese Verzeichnisse können Sie in der *tftpd*-Kommandozeile angeben, was in der zweiten *tftp*-Zeile des Beispiels zu erkennen ist.

Die tcpd-Zugriffs-Kontrolleinrichtung

Weil das Erweitern eines Computers um den Netzwerkzugriff viele Sicherheitsrisiken in sich birgt, sind die Anwendungen so entwickelt worden, daß sie sich vor verschiedenen Arten von Angriffen schützen können. Allerdings sind einige Sicherheitsmerkmale mangelhaft (was der RTM Internet-Wurm auf drastische Weise demonstriert hat), oder es wird nicht unterschieden zwischen einem sicheren Host, von dem Anforderungen nach einem bestimmten Service akzeptiert werden können, und einem unsicheren Host, dessen Anforderungen abgelehnt werden müssen. Die Dienste *finger* und *tftp* haben wir oben ja bereits angesprochen. Sicher würden Sie den Zugriff auf diese Dienste gerne auf »vertrauenswürdige Hosts« beschränken, was aber mit dem normalen Setup nicht möglich ist, bei dem *inetd* den Dienst entweder allen Clients anbietet, oder keinem.

Ein für solche Fälle nützliches Werkzeug ist *tcpd*, ein sogenannter Dämon-Wrapper. Bei TCP-Diensten, die Sie überwachen oder sichern wollen, wird es anstelle des normalen Server-Programms gestartet. *tcpd* schickt eine Meldung über die Anforderungen an den *syslog*-Dämon, prüft, ob der entfernte Host diesen Dienst überhaupt benutzen darf und führt nur dann das eigentliche Server-Programm aus. Beachten Sie, daß dies bei UDP-basierten Diensten leider nicht funktioniert.(1)

Um beispielsweise den *finger*-Dämon mit einem Wrapper zu schützen, müssen Sie die entsprechende Zeile in der *inetd.conf* wie folgt ändern:

```
# finger-Dämon mit Wrapper schützen
finger   stream  tcp     nowait  root    /usr/sbin/tcpd   in.fingerd
```

Ohne eine zusätzliche Zugriffskontrolle erscheint dies für den Client wie ein ganz gewöhnliches *finger*-Setup, mit

der Ausnahme, daß alle Anforderungen in der `auth`-Einrichtung von *syslog* aufgezeichnet werden.

Die Zugriffskontrolle wird mit Hilfe der beiden Dateien */etc/hosts.allow* und */etc /hosts.deny* implementiert. Sie enthalten Einträge, die den Zugriff auf bestimmte Dienste und Hosts erlauben bzw. sperren. Erhält *tcpd* eine Anforderung für einen Dienst wie *finger* von einem Client-Host namens biff.foobar.com, durchsucht es *hosts.allow* und *hosts.deny* (in dieser Reihenfolge) nach einem Eintrag, bei dem sowohl der Dienst als auch der Client übereinstimmen. Wird ein passender Eintrag in *hosts.allow* gefunden, wird der Zugriff freigegeben, gleichgültig, ob es noch einen Eintrag in *hosts.deny* gibt. Wird eine Übereinstimmung in *hosts.deny* gefunden, wird die Anforderung abgewiesen, und die Verbindung wird unterbrochen. Wird überhaupt kein passender Eintrag gefunden, wird die Anforderung ebenfalls akzeptiert.

Die Einträge in den Zugriffsdateien sehen wie folgt aus:

```
servicelist: hostlist [:shellcmd]
```

servicelist ist eine Liste mit Servicenamen aus */etc/services* oder das Schlüsselwort `ALL`. Um alle Dienste außer *finger* und *tftp* zu akzeptieren, müssen Sie »`ALL EXCEPT finger, tftp`« eingeben.

`hostlist` ist eine Liste mit Hostnamen oder IP-Adressen oder den Schlüsselwörtern `ALL`, `LOCAL` oder `UNKNOWN`. `ALL` steht für alle Hosts, während `LOCAL` nur Hostnamen vergleicht, die keinen Punkt enthalten.(2) `UNKNOWN` gilt für jeden Host, dessen Name oder Adresse durch einen Lookup nicht ermittelt werden konnte. Ein mit einem Punkt beginnender Name steht für eine Domain und wählt alle Hosts aus, die zu dieser Domain gehören. Zum Beispiel würde der Eintrag .foobar.com das System biff.foobar.com akzeptieren. IP-Netzwerk-Adressen und Subnetz-Nummern werden ebenfalls unterstützt.

Um den Zugriff auf *finger* und *tftp* für alle Hosts, mit Ausnahme der lokalen Hosts, zu unterbinden, müssen Sie den folgenden Eintrag in die */etc/hosts.deny* eintragen, wobei */etc/hosts.allow* nicht verändert wird:

```
in.tftpd, in.fingerd: ALL EXCEPT LOCAL, .Ihre.Domain
```

Das optionale Feld *`shellcmd`* kann einen Shell-Befehl enthalten, der ausgeführt wird, wenn der Vergleich erfolgreich war. Dies ist sinnvoll, wenn Sie zusätzliche Fallen einbauen wollen, um mögliche Angreifer zu enttarnen:

```
in.ftpd: ALL EXCEPT LOCAL, .vbrew.com : \
      echo "request from %d@%h: >> /var/log/finger.log; \
      if [ %h != "vlager.vbrew.com:" ]; then \
          finger -l @%h >> /var/log/finger.log \
      fi
```

Die Argumente `%h` und `%d` werden von *tcpd* durch den Client-Hostnamen und den Servicenamen ersetzt. Details entnehmen Sie bitte der Manpage *hosts_access(5)*.

Die Dateien services und protocols

Die Portnummer, über die einige der »Standarddienste« angeboten werden, sind im »Assigned Numbers RFC« definiert. Um es Server- und Client-Programmen zu ermöglichen, die Servicenamen in diese Nummern umzuwandeln, ist zumindest einTeil dieser Liste auf jedem Host vorhanden und wird in einer Datei namens */etc/services* gespeichert. Ein Eintrag setzt sich aus den folgenden Feldern zusammen:

```
Service Port/Protokoll   [Aliase]
```

Service gibt den Namen des Dienstes an, `Port` beschreibt, auf welchem Port dieser Dienst angeboten wird, und `Protokoll` definiert das zu verwendende Transportprotokoll. Üblicherweise ist dies entweder `udp` oder `tcp`. Ein Dienst kann auch für mehr als ein Protokoll angeboten werden. Verschiedene Dienste können denselben Port benutzen, solange die Protokolle verschieden sind. Im Feld `Aliase` können Sie alternative Namen für denselben Service definieren.

Normalerweise müssen Sie die Services-Datei nicht ändern, die mit der Netzwerk-Software für Ihr Linux-System geliefert wird. Dennoch wollen wir Ihnen in Beispiel 9--2 einen kleinen Ausschnitt dieser Datei zeigen.

Beispiel 9-2. Ausschnitt aus /etc/services (Beispiel)

```
# Services-Datei:
#
# bekannte Dienste:
echo            7/tcp                     # Echo
echo            7/udp                     #
discard         9/tcp  sink null          # Discard
discard         9/udp  sink null          #
daytime        13/tcp                     # Daytime
daytime        13/udp                     #
chargen        19/tcp  ttytst source      # Character Generator
chargen        19/udp  ttytst source      #
ftp-data       20/tcp                     # File Transfer Protocol (Data)
ftp            21/tcp                     # File Transfer Protocol (Control)
telnet         23/tcp                     # Virtual Terminal Protocol
smtp           25/tcp                     # Simple Mail Transfer Protocol
nntp          119/tcp  readnews           # Network News Transfer Protocol
#
# UNIX services
exec          512/tcp                     # rexecd (BSD)
biff          512/udp  comsat             # Mail-Benachrichtigung
login         513/tcp                     # remote Login
who           513/udp  whod               # who und uptime (remote)
shell         514/tcp  cmd                # Befehl (remote) ohne Paßworteingabe
syslog        514/udp                     # System-Logging (remote)
printer       515/tcp  spooler            # Druckspooler (remote)
route         520/udp  router routed      # Router-Informationsprotokoll
```

Zum Beispiel wird der Dienst *echo* sowohl für TCP als auch für UDP über Port 7 angeboten. Port 512 wird für zwei verschiedene Dienste verwendet: für die Programmausführung auf entfernten Systemen mittels *rexec* (TCP) und für den COMSAT-Dämon (UDP), der Benutzer über neu eingegangene Post informiert.

Analog zur *services*-Datei benötigt die Netzwerk-Bibliothek noch eine Möglichkeit, Protokollnamen (beispielsweise die in der Services-Datei verwendeten) in Protokollnummern zu übersetzen, die der IP-Layer auf dem anderen Host versteht. Dies wird mit Hilfe der Datei */etc/protocols* erreicht. Sie besteht aus einem Eintrag pro Zeile, der den Protokollnamen und die zugewiesene Nummer enthält. Daß Sie sich mit dieser Datei auseinandersetzen müssen, ist sogar noch unwahrscheinlicher als bei */etc/services*. Ein Beispiel ist in Beispiel 9--3 zu sehen.

Beispiel 9-3. Ausschnitt aus /etc/protocols (Beispiel)

```
#
```

```
# Internet-Protokolle (IP)
#
ip      0       IP              # Internet-Protokoll, Pseudo-Protokollnummer
icmp    1       ICMP            # Internet Control Message Protocol
igmp    2       IGMP            # Internet Group Multicast Protocol
tcp     6       TCP             # Transmission Control Protocol
udp     17      UDP             # User Datagram Protocol
raw     255     RAW             # RAW IP-Interface
```

Remote Procedure Call

Ein sehr allgemeiner Mechanismus für Client/Server-Anwendungen wird von RPC, dem *Remote Procedure Call*-Paket, angeboten. RPC ist von Sun Microsystems entwickelt worden und ist eine Sammlung von Tools und Bibliotheksfunktionen. Wichtige Anwendungen, die auf RPC basieren, sind NFS, das »Network File System« und NIS, das »Network Information System«.

Ein RPC-Server kennt eine Reihe von Prozeduren, die ein Client aufrufen kann, indem er eine RPC-Anforderung zusammen mit den benötigten Parametern an den Server schickt. Der Server führt die Prozedur im Auftrag des Client aus und liefert das Ergebnis zurück, wenn es eines gibt. Um maschinenunabhängig zu sein, werden alle zwischen dem Client und dem Server ausgetauschten Daten vom Sender in das sogenannte XDR-Format (*External Data Representation*) um- und vom Empfänger wieder in die lokale Repräsentation zurückgewandelt. Sun hat RPC großzügigerweise in die Public Domain freigegeben. Es wird in einer Reihe von RFCs beschrieben.

Manchmal führen Verbesserungen an einer RPC-Anwendung zu Inkompatibilitäten im Interface der Prozeduraufrufe. Natürlich würde ein einfaches Wechseln des Servers alle Anwendungen zum Absturz bringen, die noch das ursprüngliche Verhalten erwarten. Darum verfügt jedes RPC-Programm über eine Versionsnummer, die normalerweise bei 1 beginnt und bei jeder neuen RPC-Version erhöht wird. Häufig bietet ein Server mehrere Versionen gleichzeitig an, und die Clients geben durch die Versionsnummer in ihren Anforderungen an, welche Implementierung des Dienstes verwendet werden soll.

Die Netzwerk-Kommunikation zwischen RPC-Servern und -Clients ist etwas ungewöhnlich. Ein RPC-Server bietet eine Sammlung von Prozeduren an, die als *Programm* bezeichnet wird und durch eine *Programm-Nummer* eindeutig definiert ist. Eine Liste, die Servicenamen auf Programm-Nummern abbildet, wird normalerweise in der Datei */etc/rpc* gespeichert. Einen Ausschnitt dieser Datei sehen Sie in Beispiel 9--4.

Beispiel 9-4. Ausschnitt aus /etc/rpc

```
#
# /etc/rpc -- verschiedene RPC-basierte Dienste
#
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rstat_svc rup perfmeter
rusersd         100002  rusers
nfs             100003  nfsprog
ypserv          100004  ypprog
mountd          100005  mount showmount
ypbind          100007
walld           100008  rwall shutdown
yppasswdd       100009  yppasswd
bootparam       100026
ypupdated       100028  ypupdate
```

Bei TCP/IP-Netzwerken wurden die Autoren von RPC mit dem Problem konfrontiert, Programm-Nummern auf Netzwerkdienste abzubilden. Sie entschieden, daß jeder Server sowohl einen TCP- als auch einen UDP-Port für jedes Programm und jede Version bereitstellen soll. Normalerweise verwenden RPC-Anwendungen UDP, um Daten zu übertragen, und greifen nur dann auf TCP zurück, wenn die zu transportierenden Daten nicht in ein einziges UDP-Datagramm passen. Es gibt allerdings einige wenige Dienste, die nur eines der beiden Transportprotokolle anbieten.

Natürlich müssen Client-Programme eine Möglichkeit haben herauszufinden, auf welche Programm-Nummer ein Port abgebildet ist. Die Verwendung einer Konfigurationsdatei zu diesem Zweck wäre zu unflexibel. Weil RPC-Anwendungen keine reservierten Ports verwenden, gibt es keine Garantie, daß ein ursprünglich von unserer Datenbank-Anwendung zu verwendender Port nicht von einem anderen Prozeß besetzt wurde. Darum nimmt eine RPC-Anwendung den ersten Port, den sie kriegen kann, und teilt ihn dem sogenannten *Portmapper-Dämon* mit. Der Portmapper fungiert als Service-Vermittler für alle RPC-Servern, die auf dieser Maschine laufen. Ein Client, der einen Dienst mit einer gegebenen Programm-Nummer nutzen möchte, wird zuerst den Portmapper auf dem Host des Servers abfragen, der dann die TCP- und UDP-Portnummern zurückliefert, über die der Dienst erreicht werden kann.

Diese Methode hat den Nachteil, daß sie -- genau wie *inetd* für die normalen Berkeley-Dienste -- einen »single point of failure« darstellt. Nun ist dieser Fall noch ein wenig schlimmer, weil alle RPC-Port-Informationen verlorengehen, wenn der Portmapper stirbt. Das bedeutet üblicherweise, daß Sie alle RPC-Server manuell neu starten, oder sogar die gesamte Maschine neu hochfahren müssen.

Linux Bei Linux wird der Portmapper *rpc.portmap* genannt und ist in */usr/sbin* zu finden. Sie müssen nur sicherstellen, daß er von *rc.inet2* aus gestartet wird; weitere Konfigurationsarbeiten sind nicht notwendig.

Konfiguration der r-Befehle

Es gibt eine Reihe von Befehlen zur Ausführung von Befehlen auf entfernten (remote) Hosts. Diese sind *rlogin*, *rsh* und *rcp*. Alle führen eine Shell auf dem entfernten Host aus und erlauben dem Benutzer die Ausführung von Befehlen. Natürlich benötigt der Client einen Account auf dem Host, auf dem die Befehle ausgeführt werden sollen. Aus diesem Grund führen all diese Befehle eine Autorisierungsprozedur durch. Üblicherweise teilt der Client dem Server den Loginnamen des Benutzers mit, der wiederum ein Paßwort anfordert, das auf die übliche Weise kontrolliert wird.

Manchmal ist es wünschenswert, für bestimmte Benutzer die Autorisierungs-Prüfungen zu vereinfachen. Wenn Sie beispielsweise sehr häufig auf eine andere Maschine in Ihrem LAN zugreifen müssen, wäre es für Sie sehr angenehm, wenn Sie nicht jedesmal Ihr Paßwort eingeben müßten.

Das Deaktivieren der Autorisierung ist nur bei einer kleinen Anzahl von Hosts ratsam, deren Paßwort-Datenbanken abgeglichen sind, oder bei einer kleinen Anzahl privilegierter Benutzer, die aus administrativen Gründen auf viele Maschinen zugreifen müssen. Wenn Sie Leuten das Login auf Ihr System ohne die Angabe eines Login-ID oder eines Paßworts gewähren wollen, müssen Sie darauf achten, nicht versehentlich jemand anderem den Zugriff zu ermöglichen.

Es gibt zwei Wege, die Autorisierungs-Prüfungen für die *r*-Befehle zu deaktivieren. Zum einen kann der Superuser einigen oder allen Benutzern auf einigen oder allen Hosts (letzteres wäre eine sehr schlechte Idee) erlauben, sich einzuloggen, ohne daß nach einem Paßwort gefragt wird. Dieser Zugriff wird in einer Datei namens */etc/hosts.equiv* kontrolliert. Sie enthält eine Liste mit Host- und Benutzernamen, die mit den Benutzern auf dem

lokalen Host gleichgesetzt werden. Eine alternative Möglichkeit wäre, daß ein Benutzer anderen Anwendern auf verschiedenen Hosts den Zugriff auf seinen Account erlaubt. Dies kann in der Datei *.rhosts* im Home-Verzeichnis des Benutzers eingetragen werden. Aus Sicherheitsgründen muß diese Datei dem Benutzer oder dem Superuser gehören und darf nicht einfach ein symbolischer Link sein. Anderenfalls wird sie ignoriert.(3)

Wenn ein Client einen der *r*-Dienste anfordert, wird der entsprechende Host- und Benutzername in der Datei */etc/hosts.equiv* gesucht. Diese Prozedur wird danach in der *.rhosts* des Benutzers, bei dem der Login erfolgen soll, wiederholt. Nehmen wir beispielsweise an, daß **janet** an **gauss** arbeitet und versucht, sich in **joes** Account auf **euler** einzuloggen. Während des nachfolgenden Beispiels sprechen wir von Janet als *Client*-Benutzer und Joe als *lokalem* Benutzer. Janet gibt auf **gauss** den folgenden Befehl ein:

```
$ rlogin -l joe euler
```

Der Server durchsucht zuerst *hosts.equiv*,(4) um zu sehen, ob Janet freier Zugang gewährt werden soll. Schlägt dies fehl, wird versucht, sie in *.rhosts* im Home-Verzeichnis von joe zu finden.

Die Datei *hosts.equiv* auf **euler** sieht wie folgt aus:

```
gauss
euler
-public
quark.physics.groucho.edu     andres
```

Ein Eintrag besteht aus dem Hostnamen, dem optional ein Benutzername folgt. Besteht der Eintrag nur aus dem Hostnamen, ist allen Benutzern dieses Host der Zugang zu diesem Host ohne weitere Prüfungen erlaubt. In unserem obigen Beispiel könnte sich Janet unter ihrem Account **janet** einloggen, wenn sie von **gauss** käme. Dasselbe würde für alle anderen Benutzer, mit Ausnahme von **root**, gelten. Wenn Janet sich allerdings als **joe** einloggen würde, müßte sie, wie gewöhnlich, das Paßwort eingeben.

Wenn, wie in der letzten Zeile des obigen Beispiels, dem Hostnamen ein Benutzername folgt, wird dem Benutzer paßwortfreier Zugang zu *allen* Accounts gewährt (mit Ausnahme des root-Accounts).

Vor dem Hostnamen kann auch ein Minuszeichen stehen, wie im Eintrag **-public**. In diesem Fall wird die Autorisierung für alle Accounts auf **public** benötigt, gleichgültig, welche Rechte von einzelnen Benutzern in deren jeweiligen *.rhosts* vergeben wurden.

Das Format von *.rhosts* ist mit dem von *hosts.equiv* identisch, die Bedeutung ist aber etwas unterschiedlich. Sehen wir uns mal Joes *.rhosts* auf euler an:

```
gauss      janet
```

Der erste Eintrag erlaubt **joe** freien Zugriff, wenn er sich von **chomp.cs**. **groucho.edu** aus einloggt, hat aber keinen Einfluß auf die Rechte anderer Accounts auf **euler** oder **chomp**. Der zweite Eintrag ist eine leicht abgewandelte Variante des ersten Eintrags und erlaubt janet den freien Zugriff auf Joes Account, wenn sie sich von **gauss** aus einloggt.

Beachten Sie, daß der Hostname des Client ermittelt wird, indem die Adresse des Anrufers auf einen Namen abgebildet wird (»Reverse Mapping«). Diese Operation schlägt aber fehl, wenn Hosts dem Resolver nicht bekannt sind. Es wird erwartet, daß der Hostname des Client dem Namen in den Hostdateien in einer der folgenden Formen entspricht:

1. Der kanonische Hostname (kein Alias) entspricht dem Hostnamen in der Datei.
2. Wenn der Hostname des Client ein voll qualifizierter Domainname ist (wie er z. B. vom Resolver zurückgeliefert wird, wenn DNS läuft), und nicht genau dem Namen aus der Host-Datei entspricht, wird

letzterer um den lokalen Domainnamen ergänzt und erneut verglichen.

Fußnoten

(1)
: Entwickelt von Wietse Venema, wietse@wzv.win.tue.nl.

(2)
: Üblicherweise enthalten nur lokale Hostnamen, die über einen Lookup in /etc/hosts ermittelt wurden, keinen Punkt.

(3)
: In einer NFS-Umgebung müssen Sie ihr möglicherweise den Modus 444 zuweisen, weil der Superuser häufig nur sehr eingeschränkt auf Dateien zugreifen kann, die über NFS gemountet sind.

(4)
: Beachten Sie, daß hosts.equiv nicht durchsucht wird, wenn sich jemand als root einloggt.

Kapitel 10
Das »Network Information System«

Wenn Sie ein lokales Netzwerk betreiben, ist Ihr Ziel üblicherweise, den Benutzern eine Umgebung zur Verfügung zu stellen, die das Netzwerk transparent erscheinen läßt. Ein wichtiger Schritt in diese Richtung ist die Synchronisation aller lebenswichtigen Daten wie Account-Informationen zwischen allen Hosts. Wir haben bereits gesehen, daß für die Auflösung von Hostnamen ein leistungsfähiger Service bereitsteht -- DNS. Für andere Aufgaben gibt es keinen so spezialisierten Service. Darüber hinaus scheint die Einrichtung von DNS auf einem kleinen LAN ohne Internet-Connectivity kaum der Mühe wert zu sein.

Aus diesem Grund entwickelte Sun NIS, das *Network Information System*. NIS bietet einfache Datenbank-Zugriffseinrichtungen, die verwendet werden können, um Informationen, wie sie in den Dateien *passwd* und *groups* enthalten sind, an alle Hosts in Ihrem Netzwerk zu verteilen. So erscheint das Netzwerk als einzelnes System mit denselben Accounts auf allen Hosts. Auf ähnliche Weise können Sie NIS einsetzen, um die Informationen zu Hostnamen aus */etc/hosts* an alle Maschinen im Netzwerk zu verteilen.

NIS basiert auf RPC und besteht aus einem Server, einer Bibliothek für die Client-Seite und verschiedenen administrativen Tools. Bekannt wurde NIS unter dem Namen *Yellow Pages* (»Gelbe Seiten«), kurz YP, der heute immer noch häufig verwendet wird. Andererseits ist »Yellow Pages« ein Warenzeichen der British Telecom, was dazu führte, daß Sun den Namen fallen ließ. Aber wie die Dinge nun einmal laufen, bleiben manche Namen einfach hängen, und so lebt YP als Präfix für die Namen der meisten NIS-Befehle wie *ypserv*, *ypbind* etc. weiter.

Heutzutage ist NIS für nahezu jedes UNIX-System verfügbar, es existieren sogar freie Implementierungen. Eine davon ist die BSD-Net-2-Release, die von einer freien Referenz-Implementierung abgeleitet ist, die Sun bereitgestellt hat. Der Client-Library-Kode dieser Release ist seit langer Zeit Teil der GNU-*libc* enthalten, während die administrativen Programme erst kürzlich von Swen Thümmler(1) implementiert wurden. Ein NIS-Server aus der Referenz-Implementierung fehlt aber. Tobias Reber hat ein anderes NIS-Paket geschrieben, das alle Tools und einen Server enthält; es heißt *yps*.(2)

Momentan wird der gesamte NIS-Kode unter der Bezeichnung NYS von Peter Eriksson(3) neu geschrieben. Unterstützt wird dabei sowohl das einfache NIS als auch das von Sun stark überarbeitete NIS+. NYS stellt nicht nur eine Reihe von NIS-Tools und einen Server zur Verfügung, sondern enthält auch einen neuen Satz von Bibliotheks-Funktionen, die möglicherweise einmal in die Standard-*libc* übernommen werden. Es schließt ein neues Konfigurations-Schema zur Auflösung von Hostnamen ein, das das aktuelle Schema ersetzt, das noch *host.conf* verwendet. Die Features dieser Funktionen werden noch weiter unten erläutert.

Dieses Kapitel behandelt schwerpunktmäßig NYS und nicht so sehr die beiden anderen Pakete, die ich als den »traditionellen« NIS-Kode bezeichne. Wenn Sie eines dieser Pakete verwenden wollen, könnten die Anweisungen in diesem Kapitel ausreichen oder auch nicht. Für weitere Informationen seien Sie auf die entsprechende NIS-Literatur verwiesen.

Zur Zeit befindet sich NYS immer noch in der Entwicklung. Darum erkennen die Standard-Linux-Utilities wie die Netzwerk-Programme oder das *login*-Programm noch nicht das neue NYS-Konfigurationsschema. Trotzdem wurde NYS in die Hauptbibliothek *libc* (seit Version 4.6) integriert, so daß Sie Ihre eigene C-Bibliothek mit NYS-Unterstützung anstelle des traditionellen NIS-Kodes aufbauen können. Standard ist aber immer noch der traditionelle NIS-Kode. Das GNU-Projekt scheint auch daran interessiert zu sein, NYS in seine offiziellen GNU-*libc* aufzunehmen, von der die Linux-Bibliothek abstammt. (Informationen zum Kompilieren einer C-Bibliothek mit NYS-Unterstützung finden Sie in der Datei *README.nys*, die der Library-Quelldistribution beiliegt.)

Vertraut werden mit NIS

NIS hält seine Datenbank-Informationen in sogenannten *Maps* (etwa: Zuordnungen), die Key/Value-Paare enthalten. Die Maps werden auf einem zentralen Host gespeichert, auf dem der NIS-Server läuft. Clients können die Informationen dann von diesem Server über verschiedene RPC-Calls abrufen. Häufig werden Maps in DBM-Dateien gespeichert.(4)

Die Maps selbst werden normalerweise aus Master-Textdateien wie */etc/hosts* oder */etc/passwd* generiert. Aus manchen Dateien werden mehrere Maps generiert, jeweils eine pro Suchschlüssel. Zum Beispiel können Sie die Datei *hosts* nach einem Hostnamen ebenso wie nach einer IP-Adresse durchsuchen. Entsprechend werden zwei NIS-Maps daraus erzeugt: *hosts.byname* und *hosts.byaddr*. Tabelle 10--1 enthält eine Liste der gängigen Maps sowie der Dateien, aus denen sie generiert werden.

Master-Datei	**Map(s)**
/etc/hosts	*hosts.byname, hosts.byaddr*
/etc/networks	*networks.byname, networks.byaddr*
/etc/passwd	*passwd.byname, passwd.byuid*
/etc/group	*group.byname, group.bygid*
/etc/services	*services.byname, services.bynumber*

```
/etc/rpc          rpc.byname, rpc.bynumber
/etc/protocols    protocols.byname, protocols.bynumber
/usr/lib/aliases  mail.aliases
-------------------------------------------------------
```

Tabelle 10.1: *Einige Standard-NIS-Maps und die entsprechenden Dateien*

In dem einen oder anderen NIS-Paket könnten auch andere Dateien und Maps unterstützt werden. Diese enthalten in der Regel Informationen zu in diesem Buch nicht behandelten Programmen. Dazu gehört beispielsweise die *bootparams*-Map, die vom *bootparamd*-Server von Sun verwendet wird.

Bei manchen Maps werden normalerweise »Spitznamen« (*Nicknames*) verwendet, die kürzer und darum auch einfacher einzugeben sind. Beachten Sie, daß diese Spitznamen nur von *ypcat* und *ypmatch* interpretiert werden können, zwei Tools, mit denen Sie Ihre NIS-Konfiguration prüfen können. Um eine Übersicht aller von diesen Programmen verstandenen Spitznamen zu erhalten, können Sie den folgenden Befehl ausführen:

```
$ ypcat -x
NIS map nickname translation table:
        "passwd" -> "passwd.byname"
        "group" -> "group.byname"
        "networks" -> "networks.byaddr"
        "hosts" -> "hosts.byname"
        "protocols" -> "protocols.bynumber"
        "services" -> "services.byname"
        "aliases" -> "mail.aliases"
        "ethers" -> "ethers.byname"
        "rpc" -> "rpc.bynumber"
        "netmasks" -> "netmasks.byaddr"
        "publickey" -> "publickey.byname"
        "netid" -> "netid.byname"
        "passwd.adjunct" -> "passwd.adjunct.byname"
        "group.adjunct" -> "group.adjunct.byname"
        "timezone" -> "timezone.byname"
```

Das NIS-Serverprogramm wird traditionell *ypserv* genannt. Bei einem durchschnittlichen Netzwerk reicht ein Server üblicherweise aus, große Netzwerke verwenden mehrere davon auf unterschiedlichen Maschinen und Segmenten des Netzwerks, um die Last der Server und Router zu verringern. Die Server werden synchronisiert, indem einer als *Master-Server* und die anderen als *Slave-Server* eingerichtet werden. Die Maps werden nur auf dem Host des Master-Servers erzeugt. Von dort aus werden sie an alle Slaves verteilt.

Wir haben uns bisher nur sehr vage über »Netzwerke« unterhalten. Tatsächlich gibt es unter NIS ein eindeutiges Konzept, das ein Netzwerk als Sammlung von Hosts definiert, die sich einen Teil ihrer System-Konfigurationsdaten über NIS teilen: die *NIS-Domain*. Leider haben NIS-Domains überhaupt nichts mit den Domains gemeinsam, die wir bei DNS kennengelernt haben. Um in diesem Kapitel also Verwechslungen zu vermeiden, werde ich auch immer angeben, von welchem Typ ich gerade spreche.

Eine NIS-Domain hat nur eine rein administrative Funktion. Mit Ausnahme der gemeinsamen Nutzung von Paßwörtern zwischen allen Maschinen der Domain ist sie für den Benutzer größtenteils unsichtbar. Darum ist der einer NIS-Domain vergebene Name nur für die Administratoren von Bedeutung. In der Regel kann jeder Name verwendet werden, solange er sich von anderen NIS-Domainnamen in Ihrem lokalen Netz unterscheidet. Beispielsweise könnten die Administratoren der virtuellen Brauerei sich entschließen, zwei NIS-Domains aufzubauen, eine für die Brauerei und eine für die Winzerei. Als Namen werden brewery und winery vereinbart. Ein anderes häufig verwendetes Schema ist die Verwendung des DNS-Domainnamens auch für NIS. Sie können den NIS-Domainnamen mit dem Befehl *domainname* setzen. Rufen Sie ihn ohne Argumente auf, wird der aktuelle NIS-Domainname ausgegeben. Wenn Sie den Namen neu setzen wollen, müssen Sie Superuser werden und folgendes eingeben:

```
# domainname brewery
```

NIS-Domains bestimmen, welchen NIS-Server eine Anwendung abfragt. Zum Beispiel sollte das *login*-Programm auf einem Host der Winzerei natürlich nur den NIS-Server der Winzerei (oder einen davon, falls es mehrere gibt) nach den Paßwort-Informationen des Benutzers abfragen. Eine Anwendung auf einem Host der Brauerei sollte entsprechend die Server der Brauerei verwenden.

Ein Geheimnis muß noch gelüftet werden, nämlich wie ein Client herausfindet, auf welchen Server er zugreifen muß. Die einfachste Lösung wäre eine Konfigurations-Datei, die den Namen des Host enthält, auf dem sich der Server befindet. Andererseits ist diese Lösung ziemlich unflexibel, weil sie es Clients nicht erlaubt, abhängig von ihrer Verfügbarkeit auf andere Server (innerhalb derselben Domain natürlich) zuzugreifen. Darum benötigen traditionelle NIS-Implementierungen einen speziellen Dämon namens *ypbind*, um den passenden NIS-Server ihrer NIS-Domain zu ermitteln. Bevor Sie also in der Lage ist, NIS-Anfragen durchzuführen, ermittelt eine Anwendung zuerst über *ypbind*, welcher Server zu verwenden ist.

ypbind sucht nach Servern, indem es Anforderungen an alle Stationen im lokalen IP-Netzwerk verschickt. Der erste, der antwortet, wird als der schnellste betrachtet und in allen weiteren NIS-Anfragen eingesetzt. Nach einer gewissen Zeit, oder falls der Server nicht mehr erreichbar ist, sucht *ypbind* erneut nach aktiven Servern.

Das Fragwürdige an dynamischer Bindung ist nur, daß sie nur selten benötigt wird und daß sie ein Sicherheitsproblem darstellt: *ypbind* vertraut dem Antworter blind, sei es ein einfacher NIS-Server oder ein böser Eindringling. Dies ist natürlich besonders gefährlich, wenn Sie Ihre Paßwort-Datenbanken über NIS verwalten. Um sich davor zu schützen, wird *ypbind* von der Linux-NIS-Bibliothek per Standardeinstellung *nicht* verwendet; der Hostname des Servers wird vielmehr aus einer Konfigurations-Datei gelesen.

NIS verglichen mit NIS+

NIS und NIS+ haben nur wenig mehr als den Namen und ein Ziel gemeinsam. NIS+ ist auf völlig andere Weise strukturiert. Anstelle eines flachen Namensraums mit unzusammenhängenden NIS-Domains verwendet es ähnlich wie DNS einen hierarchischen Namensraum. Statt Maps werden sogenannte *Tabellen* verwendet, die aus mehreren Zeilen und Spalten bestehen. Jede Zeile steht für ein Objekt in der NIS+-Datenbank, und die Spalten beschreiben die Eigenschaften der Objekte, von denen NIS+ weiß und

die es verwaltet. Jede Tabelle für eine gegebene NIS+Domain umfaßt die Tabellen seiner Parent-Domains. Darüber hinaus kann jeder Eintrag in der Tabelle einen Verweis auf eine andere Tabelle enthalten. Auf diese Weise ist es möglich, Informationen auf unterschiedlichste Arten zu strukturieren.

Das traditionelle NIS verwendet die RPC-Versionsnummer 2, während NIS+ Version 3 hat. NIS+ ist bisher nicht sonderlich weit verbreitet, und ich weiß wirklich nicht besonders viel darüber. (Nun, eigentlich gar nichts.) Aus diesem Grund wird hier auch nicht weiter darauf eingegangen.

Die Client-Seite von NIS

Wenn Sie mit dem Schreiben oder Portieren von Netzwerk-Anwendungen vertraut sind, wird Ihnen aufgefallen sein, daß die meisten der oben aufgeführten NIS-Maps mit Bibliotheks-Funktionen der C-Library übereinstimmen. Um beispielsweise *passwd*-Informationen zu bekommen, nutzen Sie die Funktionen *getpwnam* und *getpwuid*, die die mit einem Benutzernamen oder Benutzer-ID verknüpften Account-Informationen zurückgeben. Unter normalen Umständen führen die Funktionen die erforderlichen Lookups in den Standard-Dateien wie */etc/passwd* durch.

Eine Implementierung dieser Funktionen, bei der NIS berücksichtigt wird, modifiziert dieses Verhalten etwas und fügt einen RPC-Aufruf ein, damit der NIS-Server nach dem Benutzernamen oder -ID sucht. In der Anwendung wird diese Operation transparent durchgeführt. Die Funktion kann die NIS-Map entweder an die Originaldatei »anhängen« oder diese komplett »ersetzen«. Natürlich handelt es sich dabei nicht um eine wirkliche Modifizierung der Datei; es sieht für die Anwendung nur so aus, als wäre die Datei ersetzt bzw. erweitert worden.

Bei traditionellen NIS-Implementierungen gab es verschiedene Konventionen darüber, welche Maps die Original-Informationen ersetzten, und welche an sie angehängt wurden. Einige, wie beispielsweise die *passwd*-Maps, benötigten Modifikationen an der *passwd*-Datei. Wurden diese nicht korrekt durchgeführt, öffneten sich aber Sicherheitslücken. Um diesem Fallstrick zu entgehen, verwendet NYS ein allgemeines Konfigurations-Schema, das bestimmt, ob und in welcher Reihenfolge eine Reihe von Client-Funktionen die Original-, NIS- oder NIS+-Dateien verwendet. Darauf wird später in diesem Kapitel noch eingegangen.

Betrieb eines NIS-Servers

Nach so viel Theorie wird es Zeit, daß wir uns bei der eigentlichen Konfigurations-Arbeit die Hände schmutzig machen. In diesem Abschnitt wird die Konfiguration eines NIS-Servers behandelt. Läuft bereits ein NIS-Server in Ihrem Netz, müssen Sie keinen eigenen einrichten. In diesem Fall können Sie diesen Abschnitt getrost überspringen.

Wenn Sie nur ein wenig mit dem Server experimentieren wollen, müssen Sie darauf achten, daß Sie keinen NIS-Domainnamen verwenden, der in Ihrem Netzwerk bereits eingesetzt wird. Das könnte die

gesamten Netzwerkdienste zum Absturz bringen und einige Leute sehr unglücklich und sehr böse machen.

Zur Zeit sind für Linux zwei NIS-Server frei verfügbar. Dies ist zum einen Tobias Rebers *yps*-Paket und zum anderen das *ypserv*-Paket von Peter Eriksson. Es spielt keine Rolle, welches Paket Sie verwenden, gleichgültig, ob Sie NYS oder den Standard NIS-Client-Kode verwenden, der in die *libc* integriert ist. Während dieses Buch geschrieben wird, scheint der Kode für die Behandlung von NIS-Slave-Servern bei *yps* vollständiger zu sein. Auf der anderen Seite löst *ypserv* ein bei NIS auftretendes Sicherheitsproblem (das später noch beschrieben wird), was *yps* nicht tut. Die Wahl hängt also von Ihren Bedürfnissen ab.

Nach der Installation des Server-Programms (*ypserv*) in */usr/sbin* sollten Sie das Verzeichnis anlegen, in dem Sie die Map-Dateien speichern wollen, die Ihr Server verteilen soll. Beim Einrichten der NIS-Domain für die brewery-Domain würden die Maps in */var/yp/brewery* untergebracht werden. Der Server prüft, ob er eine bestimmte NIS-Domain bedient, indem er nachsieht, ob ein entsprechendes Map-Verzeichnis vorhanden ist. Wenn Sie den Service für eine NIS-Domain einstellen, müssen Sie auch sicherstellen, daß das Verzeichnis entfernt wird.

Maps werden üblicherweise in DBM-Dateien gespeichert, um Lookup-Zeiten zu minimieren. Diese werden aus den Master-Dateien erzeugt, wobei ein Programm namens *makedbm* (für Tobias' Server) bzw. *dbmload* (für Peters Server) eingesetzt wird. Diese DBM-Dateien sind nicht austauschbar. Die Umwandlung einer Master-Datei in eine für *dbmload* verwendbare Form muß über *awk* oder *sed* erfolgen, was lästige Schreibarbeit bedeutet und nur schwer zu merken ist. Darum enthält Peter Erikssons *ypserv*-Paket ein Makefile (*ypMakefile*), das die ganze Arbeit für Sie erledigt. Sie sollten es unter dem Namen *Makefile* in Ihrem Map-Verzeichnis installieren und es so editieren, daß alle benötigten Maps enthalten sind. Ziemlich zu Anfang der Datei finden Sie den Eintrag `all`, der die Dienste aufführt, die *ypserv* anbieten soll. Nachfolgend die Zeile, wie Sie in der Distribution zu finden ist:

```
all: ethers hosts networks protocols rpc services passwd group netid
```

Sollen beispielsweise keine *ethers.byname*- und *ethers.byaddr*-Maps erzeugt werden, entfernen Sie einfach die Bedingung `ethers` aus dieser Regel. Um Ihr Setup zu testen, können Sie zuerst mit ein oder zwei Maps wie den *services.**-Maps beginnen.

Nachdem Sie die entsprechenden Korrekturen am *Makefile* vorgenommen haben, geben Sie einfach *make* ein, während Sie sich im Map-Verzeichnis befinden. Die Maps werden nun automatisch generiert und installiert. Sie müssen die Maps jedesmal aktualisieren, wenn Sie die Master-Dateien verändern, weil diese Änderungen sonst für das Netzwerk nicht sichtbar sind.

Der nächste Abschnitt beschreibt die Konfiguration des NIS-Client-Kodes. Wenn Ihr Setup nicht funktioniert, sollten Sie prüfen, ob Anforderungen bei Ihrem Server eingehen oder nicht. Wenn Sie *ypserv* mit der Kommandozeilen-Option *-debug* starten, erscheinen auf dem Bildschirm Debugging-Informationen zu allen eingegangenen NIS-Anforderungen sowie zu den zurückgelieferten Resultaten. Dies sollte Ihnen einen Hinweis darauf geben, wo das Problem liegt. Tobias' Server verfügt über keine solche Option.

Sicherheit von NIS-Servern



NIS hatte eine große Sicherheitslücke: Beinahe jeder im gesamten Internet konnte Ihre Paßwort-Datei lesen, was für eine beträchtliche Anzahl möglicher Eindringlinge sorgte. Wenn ein Eindringling Ihren NIS-Domainnamen und die Adresse Ihres Servers kannte, konnte er einfach eine Anforderung an die *passwd.byname*-Map schicken und erhielt die gesamten Paßwörter Ihres Systems. Ausgestattet mit einem schnellen Paßwort-Knackprogramm wie *crack* und mit einem guten Wörterbuch ist es kaum ein Problem, die Paßwörter von zumindest einigen Benutzern zu knacken.

Aus diesem Grund wurde die securenets-Option eingeführt. Sie beschränkt den Zugriff auf Ihren NIS-Server anhand der IP-Adresse oder der Netzwerknummer auf einige wenige Hosts. Die neueste Version von *ypserv* implementiert dieses Feature auf sehr bequeme Weise, indem sie die Dateien */etc/hosts.allow* und */etc/hosts.deny* verwendet, die Sie ja bereits in Kapitel 9, *Wichtige NetzwerkFeatures* kennengelernt haben.(5) Um den Zugriff beispielsweise auf die Hosts innerhalb der Brauerei zu beschränken, würden deren Administratoren die folgende Zeile in *hosts.allow* aufnehmen:

```
ypserv: 172.16.2.
```

So können alle Hosts vom IP-Netzwerk 172.16.2.0 auf den NIS-Server zugreifen. Um alle anderen Hosts auszuschließen, müßte folgender Eintrag in *hosts.deny* aufgenommen werden:

```
ypserv: ALL
```

IP-Nummern sind nicht der einzige Weg, Hosts oder Netzwerke in *hosts.allow* und *hosts.deny* anzugeben. Details entnehmen Sie bitte der *hosts_access(5)*-Manpage Ihres Systems. Aber Vorsicht: Host- oder Domainnamen können bei `ypserv`-Einträgen *nicht* verwendet werden. Wenn Sie einen Hostnamen angeben, versucht der Server ihn aufzulösen -- aber der Resolver ruft wiederum *ypserv* auf, und Sie enden in einer Endlosschleife.

Sie können auch den sicheren Portmapper anstelle der securenets-Option bei *ypserv* verwenden. Der sichere Portmapper (*portmap-3.0*)(6) verwendet ebenfalls das Schema mit *hosts.allow*, bietet dieses aber für alle RPC-Server und nicht nur für *ypserv* an. Allerdings sollten Sie aufgrund des damit verbundenen Overheads die securenets-Option und den sicheren Portmapper nicht gleichzeitig verwenden.

Mit NYS einen NIS-Client einrichten

Den Rest dieses Kapitels verbringen wir mit der Konfiguration eines NIS-Client.

Im ersten Schritt sollten Sie NYS mitteilen, welcher Server für den NIS-Service verwendet werden soll. Den Namen des Servers können Sie in die Konfigurations-Datei */etc/yp.conf* eintragen. Eine sehr einfache Datei für einen Host des Winzerei-Netzwerks würde wie folgt aussehen:

```
# yp.conf -- YP-Konfiguration für NYS-Library.
#
```

```
domainname winery
server     vbardolino
```

Die erste Zeile teilt Ihrem Host mit, daß er zur NIS-Domain winery gehört. Fehlt dieser Eintrag, verwendet NYS den Domainnamen, den Sie Ihrem System mit dem Befehl *domainname* zugewiesen haben. Die `server`-Zeile benennt den zu verwendenden NIS-Server. Natürlich muß die zu vbardolino gehörende IP-Adresse in *hosts* eingetragen sein. Alternativ können Sie auch direkt die IP-Adresse in der `server`-Anweisung angeben.

In der oben aufgeführten Form weist der `server`-Befehl NYS an, den angegebenen Server zu verwenden, unabhängig von der aktuellen NIS-Domain. Wenn Sie sich aber mit Ihrem Rechner häufig in unterschiedlichen Domains bewegen, wollen Sie die Informationen zu den verschiedenen Domains ebenfalls in Ihrer *yp.conf* festhalten. Sie können Informationen zu Servern verschiedener NIS-Domains in *yp.conf* speichern, indem Sie den NIS-Domainnamen im `server`-Befehl mit angeben. Für einen Laptop könnte die obige Beispieldatei etwa wie folgt aussehen:

```
# yp.conf -- YP-Konfiguration für NYS-Library.
#
server vbardolino winery
server vstout     brewery
```

Damit können Sie den Laptop in jeweils einer von beiden Domains betreiben, indem Sie die gewünschte NIS-Domain einfach mit dem Befehl *domainname* während der Bootphase setzen. Nachdem Sie die grundlegende Konfigurationsdatei erzeugt und sichergestellt haben, daß sie allgemein lesbar ist, sollten Sie den ersten Test durchführen, um zu prüfen, ob Sie sich an Ihren Server anbinden können. Wählen Sie eine Map wie *hosts.byname*, von der Sie sicher sind, daß Ihr Server sie auch verteilt, und versuchen Sie mit Hilfe des *ypcat*-Utilities auf sie zuzugreifen. *ypcat* sollte wie alle anderen administrativen NIS-Tools in */usr/sbin* zu finden sein.

```
# ypcat hosts.byname
172.16.2.2      vbeaujolais.vbrew.com    vbeaujolais
172.16.2.3      vbardolino.vbrew.com     vbardolino
172.16.1.1      vlager.vbrew.com         vlager
172.16.2.1      vlager.vbrew.com         vlager
172.16.1.2      vstout.vbrew.com         vstout
172.16.1.3      vale.vbrew.com           vale
172.16.2.4      vchianti.vbrew.com       vchianti
```

Die Ausgabe sollte mit dem oben Gezeigten vergleichbar sein. Erhalten Sie statt dessen die Fehlermeldung »`Can't bind to server which serves domain`«, dann existiert für den von Ihnen gesetzten NIS-Domainnamen entweder kein passender Server, oder der Server ist aus irgendeinem Grund nicht erreichbar. Im letzteren Fall sollten Sie mit *ping* sicherstellen, daß der Host da ist. Daß auch tatsächlich ein NIS-Server auf diesem Host läuft, können Sie mit dem Befehl *rpcinfo* überprüfen, der Ihnen folgendes Resultat zurückliefern sollte:

```
# rpcinfo -u serverhost ypserv
program 100004 version 2 ready and waiting
```

Die Wahl der richtigen Maps

Wenn Sie sicher sind, daß Sie den NIS-Server erreichen, müssen Sie entscheiden, welche Konfigurations-Dateien durch NIS-Maps ersetzt oder erweitert werden sollen. Üblicherweise wollen Sie NIS-Maps bei den Host- und Paßwort-Lookup-Funktionen benutzen. Das letztere ist besonders nützlich, wenn Sie den BIND-Service nicht verwenden. Durch den Paßwort-Lookup können sich alle Benutzer an jedem System innerhalb der NIS-Domain einloggen. Das geht normalerweise einher mit der Nutzung eines zentralen */home*-Verzeichnisses zwischen allen Hosts über NFS. Die Paßwort-Map wird im nächsten Abschnitt ausführlich behandelt.

Andere Maps wie *services.byname* bieten Ihnen keine solch großen Möglichkeiten, sparen Ihnen aber einiges an Editierarbeit. *services.byname* ist besonders nützlich, wenn Sie Netzwerk-Anwendungen installieren, die einen Servicenamen verwenden, der nicht in der standardmäßigen *services*-Datei steht.

Normalerweise wollen Sie die Wahl haben, ob eine Lookup-Funktion die lokalen Dateien oder den NIS-Server verwendet. Bei NYS können Sie die Reihenfolge festlegen, in der auf diese Dienste zugegriffen wird. Kontrolliert wird dies durch die Datei */etc/nsswitch.conf*, die für *Name Service Switch* steht, aber natürlich nicht nur auf den Name-Service beschränkt ist. Für jede der von NYS unterstützten Lookup-Funktionen enthält sie eine Zeile, die den Namen des zu verwendenden Dienstes enthält.

Die richtige Reihenfolge der Dienste ist vom Typ der Daten abhängig. Es ist unwahrscheinlich, daß die *services.byname*-Map Einträge enthält, die sich von der lokalen *services*-Datei unterscheiden; es kann aber sein, daß sie zusätzliche Einträge enthält. Es ist also durchaus sinnvoll, zuerst die lokalen Dateien abzufragen und NIS nur dann zu nutzen, wenn der Servicename nicht gefunden werden konnte. Andererseits können sich Informationen zu Hostnamen sehr häufig verändern, so daß DNS oder NIS die aktuellsten Informationen haben sollten, während die lokale *hosts* nur als Backup verwendet wird, falls DNS und NIS nicht funktionieren sollten. In diesem Fall würden Sie die lokalen Dateien also zuletzt verwenden.

Das folgende Beispiel zeigt, wie Sie *gethostbyname*, *gethostbyaddr* und *getservbyname* zwingen können, zuerst in NIS und DNS nachzusehen, bevor auf die lokale *hosts*-Datei zugegriffen wird. Die aufgeführten Dienste werden nacheinander aufgerufen. Ist eine Suche erfolgreich, wird das Ergebnis zurückgegeben, ansonsten wird der nächste Service ausprobiert.

```
# /etc/nsswitch.conf (ein kleines Beispiel)
#
hosts:     nis dns files
services:  files nis
```

Eine komplette Liste aller Dienste, die mit einem Eintrag in *nsswitch.conf* verwendet werden können, ist nachfolgend aufgeführt. Welche Maps, Dateien, Server und Objekte tatsächlich angefordert werden, ist vom Namen des Eintrags abhängig.

nisplus oder nis+

Benutze den NIS+-Server für diese Domain. Der Name des Servers wird aus der Datei

/etc/nis.conf ermittelt.

nis

Benutze den aktuellen NIS+-Server für diese Domain. Der Name des angeforderten Servers wird, wie im vorhergehenden Abschnitt beschrieben, in *yp.conf* konfiguriert. Beim `hosts`-Eintrag werden die Maps *hosts.byname* und *hosts.byaddr* abgefragt.

dns

Benutze den DNS-Nameserver. Dieser Service-Typ ist nur zusammen mit dem `hosts`-Eintrag sinnvoll. Die abgefragten Name-Server werden weiter aus der Standarddatei *resolv.conf* ermittelt.

files

Benutze die lokale Datei, wie z. B. */etc/hosts* für den `hosts`-Eintrag.

dbm

Suche die Informationen in DBM-Dateien im Verzeichnis */var/dbm*. Der für die Datei verwendete Name ist der der entsprechenden NIS-Map.

Momentan unterstützt NYS die folgenden *nsswitch.conf*-Einträge: `hosts`, `networks`, `passwd`, `group`, `shadow`, `gshadow`, `services`, `protocols`, `rpc` und `ethers`. Weitere werden wohl folgen.

Beispiel 10--1 zeigt ein vollständigeres Beispiel, das ein weiteres Feature von *nsswitch.conf* einführt. Das Schlüsselwort `[NOTFOUND=return]` im `hosts`-Eintrag weist NYS an, die Suche abzubrechen, wenn der benötigte Eintrag in der NIS- oder DNS-Datenbank nicht gefunden werden kann. Das bedeutet, daß NYS die Suche in lokalen Dateien nur aufnimmt, wenn Anforderungen an die NIS- oder DNS-Server aus einem anderen Grund fehlgeschlagen sind. Die lokalen Dateien werden dann nur während der Bootphase und als Backup (falls der NIS-Server nicht erreichbar ist) benutzt.

Beispiel 10-1. nsswitch.conf-Beispieldatei

```
# /etc/nsswitch.conf (Beispieldatei)
#
hosts:      nis dns [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files
services:   files nis
protocols:  files nis
rpc:        files nis
```

Verwendung von passwd- und group-Maps

Eine der Hauptanwendungen von NIS ist die Synchronisation von Benutzer- und Account-Informationen auf allen Hosts in einer NIS-Domain. Daher haben Sie üblicherweise nur eine kleine lokale */etc/passwd*-Datei, an die die Domain-weiten Informationen aus den NIS-Maps angehängt werden. Allerdings ist die Aktivierung von NIS-Lookups für diesen Service in *nsswitch.conf* bei weitem nicht genug.

Wenn Sie sich auf die von NIS verteilten Paßwort-Informationen verlassen, müssen Sie zuerst

sicherstellen, daß die numerischen Benutzer-IDs aller in Ihrer lokalen *passwd*-Datei stehenden Benutzer mit den Vorstellungen des NIS-Servers von Benutzer-IDs übereinstimmen. Sie benötigen dies auch für andere Zwecke, wie dem Mounten von NFS-Verzeichnissen von anderen Hosts in Ihrem Netzwerk.

Wenn sich numerische IDs in */etc/passwd* oder */etc/group* von denen in den Maps unterscheiden, müssen Sie zuerst die Besitzrechte für alle Dateien korrigieren, die diesem Benutzer gehören. Zuerst sollten Sie alle UIDs und GIDs in *passwd* und *group* auf die neuen Werte setzen, alle Dateien suchen, die den gerade geänderten Benutzern gehören, und schließlich deren Besitzrechte entsprechend ändern. Nehmen wir einfach an, news hätte die Benutzer-ID 9 und okir hätte die Benutzer-ID 103, die beide durch einen anderen Wert ersetzt wurden. Sie können dann die folgenden Befehle eingeben:

```
 # find / -uid   9 -print >/tmp/uid.9
 # find / -uid 103 -print >/tmp/uid.103
 # cat /tmp/uid.9   | xargs chown news
 # cat /tmp/uid.103 | xargs chown okir
```

Es ist wichtig, daß Sie diese Befehle mit der neu installierten *passwd*-Datei aufrufen und daß Sie alle Dateinamen ermitteln, deren Besitzerrechte Sie ändern. Zum Aktualisieren der Gruppen-Besitzerrechte verwenden Sie einen vergleichbaren Befehl.

Nachdem Sie dies erledigt haben, stimmen die numerischen UIDs und GIDs auf Ihrem System mit denen auf allen anderen Hosts in Ihrer NIS-Domain überein. Der nächste Schritt ist, einige Konfigurations-Zeilen in die *nsswitch.conf* einzutragen, die die NIS-Lookups für Benutzer- und Gruppeninformationen aktivieren:

```
# /etc/nsswitch.conf -- passwd and group treatment
passwd: nis files
group:  nis files
```

Dies hat Einfluß darauf, wo der *login*-Befehl und all seine Freunde nach Benutzerinformationen Ausschau halten. Versucht sich ein Benutzer einzuloggen, fragt *login* zuerst die NIS-Maps ab; erst wenn dies fehlschlägt, wird auf die lokalen Dateien zurückgegriffen. Üblicherweise entfernen Sie alle Benutzer aus Ihren lokalen Dateien und lassen nur Einträge für root und Accounts wie mail stehen. Der Grund dafür liegt darin, daß einige wichtige Tasks UIDs auf Benutzernamen abbilden müssen und umgekehrt. Zum Beispiel könnten administrative *cron*-Jobs den *su*-Befehl ausführen, um kurzzeitig news zu werden, oder das UUCP-Subsystem könnte einen Statusreport über E-Mail übertragen. Besitzen news und uucp keinen Eintrag in der lokalen *passwd*-Datei, schlagen diese Jobs während eines NIS-Ausfalls kläglich fehl.

Linux Nun gibt es an dieser Stelle aber zwei große Vorbehalte. Zum einen funktioniert das bisher beschriebene Setup nur für Login-Suites, die keine Shadow-Paßwörter wie die aus dem *util-linux*-Paket benutzen. Die Komplikationen bei der Verwendung von Shadow-Paßwörtern mit NIS werden später noch behandelt. Zum anderen sind die Login-Befehle nicht die einzigen, die auf die *passwd*-Datei zugreifen -- sehen Sie sich den *ls*-Befehl an, den die meisten Leute fast dauernd benutzen. Wenn Sie ein ausführliches Inhaltsverzeichnis anfordern, gibt *ls* die symbolischen Namen der Benutzer- und

Gruppenbesitzer einer Datei aus. Das bedeutet, daß für jede UID und GID, der *ls* begegnet, der NIS-Server einmal abgefragt werden muß. Das kann die Dinge ganz beträchtlich verlangsamen, wenn das lokale Netz überlastet ist. Noch schlimmer sieht es aus, wenn der NIS-Server sich nicht in demselben physikalischen Netzwerk befindet, so daß Datagramme über einen Router laufen müssen.

Aber das ist noch nicht alles. Stellen Sie sich vor, was passiert, wenn eine Benutzerin ihr Paßwort ändern möchte. Normalerweise würde sie *passwd* aufrufen, das das neue Paßwort einliest und die lokale *passwd*-Datei aktualisiert. Das ist bei NIS unmöglich, weil die Datei lokal nicht mehr vorhanden ist. Daß sich Benutzer in den NIS-Server einloggen, wenn sie das Paßwort ändern wollen, ist auch keine Lösung. NIS stellt aus diesem Grund ein Programm namens *yppasswd* als Ersatz für *passwd* bereit, das dieselbe Aufgabe übernimmt, wenn NIS präsent ist. Um das Paßwort auf dem Server-Host zu ändern, ruft es den *yppasswdd*-Dämon auf diesem Host über RPC auf und versorgt ihn mit der aktualisierten Paßwort-Information. Normalerweise ersetzen Sie das normale Programm durch *yppasswd*, beispielsweise durch die nachstehende Befehlsfolge:

```
# cd /bin
# mv passwd passwd.old
# ln yppasswd passwd
```

Gleichzeitig müssen Sie *rpc.yppasswdd* auf dem Server installieren und ihn von *rc.inet2* aus starten. Auf diese Weise werden die gesamten Klimmzüge von NIS vor den Benutzern versteckt.

NIS und Shadow-Paßwörter verwenden

NIS in Verbindung mit Shadow-Paßwortdateien ist eine etwas trickreiche Angelegenheit. Shadow-Paßwörter wurden eingeführt, um zu verhindern, daß normale Benutzer die Paßwörter anderer Anwender, selbst im verschlüsselten Zustand, lesen können. Andererseits zwingt NIS sie dazu, diese Paßwörter netzwerkweit verfügbar sind, was dem ursprünglichen Zweck von Shadow-Paßwörtern natürlich völlig entgegenläuft.

Momentan gibt es unter Linux keine wirkliche Lösung für dieses Dilemma. Der einzige Weg, Paßwort- und Benutzerinformationen über NIS zu verteilen, sind die Standard-*passwd.**-Maps. Wenn Sie Shadow-Paßwörter installiert haben, besteht die einfachste Möglichkeit der gemeinsamen Nutzung darin, eine ordentliche *passwd*-Datei aus */etc/shadow* zu erzeugen (mit Tools wie *pwuncov*) und aus dieser Datei die entsprechenden NIS-Maps aufzubauen.

Natürlich gibt es einige Hacks, um NIS und Shadow-Paßwörter zur selben Zeit zu verwenden, beispielsweise indem Sie eine */etc/shadow*-Datei auf jedem Host im Netzwerk installieren, während die Benutzer-IDs etc. über NIS verteilt werden. Es ist natürlich unnötig zu erwähnen, daß diese Vorgehensweise ziemlich rüde ist und dem Ziel von NIS, die Systemadministration zu vereinfachen, völlig entgegenläuft. Meiner Meinung nach ist der Grundsatz, Benutzer dazu anzuhalten »gute«

Paßwörter zu verwenden, wesentlich besser, als die Paßworte in zusätzlichen Dateien zu verstecken, die zu Kompatibilitäts-Problemen führen.

Den traditionellen NIS-Kode verwenden

Wenn Sie den Client-Kode verwenden, der momentan in der Standard-*libc* enthalten ist, läuft die Konfiguration eines NIS-Client etwas anders ab. Zuerst unterstützt der traditionelle Kode nur Maps für *hosts*-, *passwd*- und *group*-Lookups. Auch die Art und Weise, wie Informationen aus lokalen Dateien mit denen der NIS-Maps kombiniert werden, unterscheidet sich stark von dem bei NYS verwendeten Schema.

Um beispielsweise die NIS-Paßwortmaps zu verwenden, müssen Sie die folgende Zeile irgendwo in Ihrer */etc/passwd*-Map eintragen:

```
 +:*:0:0:::
```

Dies markiert die Stelle, an der die Paßwort-Lookup-Funktionen die NIS-Maps »einfügen«. Das Einfügen einer ähnlichen Zeile (ohne die letzten beiden Doppelpunkte) in */etc/group* erledigt dasselbe für die *group.**-Maps.

Um die verschiedenen von Ihrem YP-Server angebotenen *hosts.**-Maps zu nutzen, ändern Sie die `order`-Zeile in Ihrer *host.conf*-Datei. Wollen Sie z. B. etwa NIS, DNS und die Datei */etc/hosts* (in dieser Reihenfolge) verwenden, müssen Sie die Zeile wie folgt ändern:

```
order yp bind hosts
```

Anders als bei NYS ist der traditionelle YP-Kode vom *ypbind*-Dämon abhängig, um einen aktiven Server für die YP-Clients zu ermitteln. *ypbind* muß während der Bootphase gestartet werden, nachdem die NIS-Domain gesetzt und der RPC-Portmapper aktiviert wurde.

Bis vor kurzem suchte *ypbind* über RPC-Broadcasts nach Servern. Wie bereits früher angesprochen, ist diese Vorgehensweise ziemlich unsicher. Darum besitzt die neueste Version der YP-Tools (aus der *yp-linux*-Distribution) nun einen *ypbind*-Dämon, der auch die Konfigurations-Datei */etc/yp.conf* unterstützt. Existiert diese Datei, wird sie nach einer oder mehreren Zeilen durchsucht, die in etwa wie folgt aussehen:

```
# yp.conf -- benenne YP-Server für ypbind.
ypserver      vbardolino
```

ypbind überprüft dann die benannten Hosts nach aktiven Servern.(7) Wenn *ypbind* keine *yp.conf*-Datei findet oder wenn keiner der genannten Server antwortet, greift es auf die althergebrachte Methode zurück und sendet RPC-Broadcasts.

Vor kurzem gab es zahlreiche Bugreports, die sich damit beschäftigten, daß NIS bei einem Fehlversuch die folgende Fehlermeldung zurückliefert:

```
clntudp_create: RPC: portmapper failure -- RPC: unable to receive
```

Dies rührt von einer inkompatiblen Art des Formats her, in dem *ypbind* die Bindungs-Informationen an die Bibliotheks-Funktionen durchreicht. Besorgen Sie sich die neuesten Quellen der NIS-Utilities und kompilieren Sie sie, dann sollte dieses Problem behoben sein.(8)

Fußnoten

(1)

Swen kann über swen@uni-paderborn.de erreicht werden. Die NIS-Clients sind als yp-linux.tar.gz von sunsite.unc.edu unter system/Network verfügbar.

(2)

Die aktuelle Version (während dieses Buch geschrieben wurde) ist yps-0.21 und kann von ftp.lysator.liu.se aus dem Verzeichnis /pub/NYS heruntergeladen werden.

(3)

Zu erreichen unter pen@lysator.liu.se.

(4)

DBM ist eine einfache Datenbank-Verwaltungsbibliothek, die eine Hash-Technik verwendet, um Suchoperationen zu beschleunigen. Vom GNU-Projekt gibt es eine freie DBM-Implementierung namens gdbm, die Teil der meisten Linux-Distributionen ist.

(5)

Um diese Option zu aktivieren, müssen Sie den Server neu kompilieren. Bitte lesen Sie die entsprechenden Anweisungen in der der Distribution beiliegenden README-Datei.Um diese Option zu aktivieren, müssen Sie den Server neu kompilieren. Bitte lesen Sie die entsprechenden Anweisungen in der der Distribution beiliegenden README-Datei.

(6)

Verfügbar über anonymous FTP von sunsite.unc.edu unter dem Verzeichnis Linux/systems/Network.

(7)

Beachten Sie, daß das zur Angabe des Servernamens verwendete Schlüsselwort nicht mit dem von NYS übereinstimmt.

(8)

Die Quellen für yp-linux können von ftp.uni-paderborn.de aus dem Verzeichnis /pub/Linux/LOCAL heruntergeladen werden.

Kapitel 11
Das Network File System

NFS, das »Network File System«, ist wahrscheinlich der bekannteste auf RPC basierende Netzwerkdienst. Er erlaubt Ihnen, auf Dateien von entfernten Hosts so zuzugreifen, wie Sie es mit lokalen Dateien tun. Dies wird durch eine Mischung aus Kernel-Funktionalität auf der Client-Seite und einem NFS-Server auf der Server-Seite realisiert. Der Dateizugriff ist für den Client völlig transparent und funktioniert für eine Reihe von Server- und Host-Architekturen.

NFS bietet eine Reihe nützlicher Features:

- Daten, auf die von allen Benutzern zugegriffen wird, können auf einem zentralen Host gehalten werden. Clients mounten dieses Verzeichnis während der Bootphase. Beispielsweise können Sie alle Benutzer-Verzeichnisse auf einem Host speichern und alle Benutzer in Ihrem Netzwerk */home* von diesem Host mounten lassen. In Verbindung mit NIS können sich die Benutzer dann an jedem System einloggen und arbeiten dennoch mit nur einem Satz von Dateien.
- Daten, die sehr große Mengen an Plattenspeicher einnehmen, können auf einem einzigen Host gehalten werden. So könnten etwa alle zu LaTeX und METAFONT gehörenden Dateien und Programme an einem Ort gespeichert und verwaltet werden.
- Administrative Daten können auf einem Host gehalten werden. Es gibt keinen Grund mehr, *rcp* zu verwenden, um eine Datei auf 20 verschiedene Maschinen zu kopieren.

Es ist nicht besonders schwer, die grundlegenden NFS-Funktionen für den Client und den Server einzurichten. Dieses Kapitel beschreibt, wie es geht.

Linux-NFS ist größtenteils die Arbeit von Rick Sladkey,(1) der den NFS-Kernel und große Teile des NFS-Servers geschrieben hat. Letzterer wurde vom *unfsd*-NFS-Server, ursprünglich entwickelt von Mark Shand, und vom *hnfs*-Harris-NFS-Server, geschrieben von Donald Becker, abgeleitet.

Sehen wir uns an, wie NFS arbeitet. Ein Client versucht, ein Verzeichnis von einem entfernten Host an sein lokales Verzeichnis zu mounten, genau wie er dies mit einem physikalischen Gerät machen würde. Allerdings ist die für ein entferntes Verzeichnis zu verwendende Syntax anders. Soll zum Beispiel das Verzeichnis */home* vom Host vlager an */users* auf vale gemountet werden, führt der Administrator auf vale den folgenden Befehl ein:(2)

```
# mount -t nfs vlager:/home /users
```

mount versucht, über RPC eine Verbindung mit dem Mount-Dämon (*mountd*) auf vlager herzustellen. Der Server überprüft, ob vale die Erlaubnis besitzt, das fragliche Verzeichnis zu mounten; wenn ja, liefert er einen Datei-Handle zurück. Dieser Handle wird bei allen weiteren Anfragen nach Dateien unter */users* verwendet.

Greift jemand über NFS auf eine Datei zu, setzt der Kernel einen RPC-Aufruf an *nfsd* (den NFS-Dämon) auf der Server-Maschine ab. Dieser Aufruf enthält den Datei-Handle, den Namen der gewünschten Datei und den Benutzer- und Gruppen-ID als Parameter. Diese werden verwendet, um die Zugriffsrechte auf die angegebene Datei zu ermitteln. Um das Lesen und Modifizieren von Dateien durch nicht-autorisierte Benutzer zu verhindern, müssen die Benutzer- und Gruppen-IDs auf beiden Hosts identisch sein.

Bei den meisten UNIX-Implementierungen wird die NFS-Funktionalität sowohl für Clients als auch für Server durch Dämonen realisiert, die auf Kernel-Ebene angesiedelt sind und während der Bootphase vom Benutzerspeicher aus gestartet werden. Dies ist auf dem Serverhost der NFS-Dämon (*nfsd*) und auf dem Client-Host der *Block I/O Dämon* (*biod*). Um den Durchsatz zu verbessern, arbeitet *biod* mit asynchroner Ein-/Ausgabe unter Gebrauch von Read-Ahead und Write-Behind. Häufig werden auch mehrere *nfsd*-Dämonen gleichzeitig verwendet.

Die NFS-Implementierung von Linux ist von daher etwas anders, als daß der Client-Kode eng mit dem VFS-Layer (»Virtual File System«) des Kernels verbunden ist und daher keine zusätzliche Kontrolle durch *biod* benötigt. Auf der anderen Seite läuft der Server vollständig im Benutzerspeicher, d. h. es ist aufgrund der damit verbundenen Synchronisations-Fragen nahezu unmöglich, mehrere Kopien des Servers zur selben Zeit laufen zu lassen. Momentan kennt Linux-NFS auch kein Read-Ahead und Write-Behind, Rick Sladkey plant aber, es eines Tages hinzuzufügen.(3)

Das größte Problem mit dem Linux-NFS-Kode ist, daß bei der Version 1.0 des Linux-Kernel nur Speicherblöcke von maximal 4 K genutzt werden können. Die Konsequenz daraus ist, daß der Netzwerk-Kode nur Datagramme übertragen kann, deren Größe auf ungefähr 3500 Byte beschränkt ist, nachdem man die Header etc. abgezogen hat. Das bedeutet, daß Übertragungen von und zu NFS-Dämonen, die auf Systemen laufen, die per Voreinstellung größere UDP-Datagramme übertragen (z. B. 8 K bei SunOS), künstlich heruntergeschraubt werden müssen. Das hat unter Umständen schmerzliche Auswirkungen auf die Performance.(4) Diese Einschränkung ist in den späten Linux-1.1-Kerneln aufgehoben worden. Der Client-Kode ist entsprechend modifiziert worden, um von diesem Vorteil Gebrauch zu machen.

NFS vorbereiten

Bevor Sie NFS benutzen können, sei es nur als Client oder als Server, müssen Sie zuerst sicherstellen, daß die entsprechende NFS-Unterstützung in den Kernel integriert (kompiliert) ist. Neuere Kernel haben dazu ein einfaches Interface im Dateisystem *proc*, nämlich die Datei */proc/filesystems*, die Sie einfach mit *cat* ausgeben können:

```
$ cat /proc/file systems
```

```
  minix
  ext2
  msdos
nodev proc
nodev  nfs
```

Fehlt `nfs` in dieser Liste, müssen Sie den Kernel neu kompilieren und dabei NFS mit einbinden. Die Konfiguration der Netzwerk-Optionen des Kernel wird im Abschnitt »Kernel-Konfiguration« in Kapitel 3 beschrieben.

Bei älteren Kerneln vor Linux 1.1 ist die einfachste Möglichkeit, um herauszufinden, ob der Kernel NFS unterstützt oder nicht, einfach auszuprobieren, ob sich ein NFS-Dateisystem mounten läßt oder nicht. Dazu können Sie ein Testverzeichnis unter */tmp* erzeugen und versuchen, ein Verzeichnis vom NFS-Server (z. B. /home) zu mounten, das dieser an Ihre Maschine exportiert:

```
# mkdir /tmp/test
# mount server:/home /tmp/test
```

Schlägt dieser Versuch mit der Meldung »`fs type nfs not supported by kernel`,« fehl, müssen Sie den Kernel mit eingebundenem NFS neu kompilieren.

Ein NFS-Volume mounten

NFS-Volumes(5) werden fast genauso gemountet wie die normalen Dateisysteme auch. Sie verwenden dabei für *mount* die folgende Syntax:

```
# mount -t nfs nfs_volume local_dir options
```

nfs_volume is given as *`remote_host:remote_dir`*. Weil diese Notation nur bei NFS-Dateisystemen gilt, können Sie die Option *-t nfs* weglassen.

Eine ganze Reihe zusätzlicher Optionen kann beim Mounten eines NFS-Volumes angegeben werden. Diese können entweder auf die Option *-o* in der Kommandozeile folgen, oder im Optionsfeld von */etc/fstab* für dieses Volume stehen. In beiden Fällen werden mehrere Optionen durch Kommata voneinander getrennt. In der Kommandozeile angegebene Optionen überschreiben immer die in *fstab*stehenden Werte.

Nachfolgend ein Beispiel-Eintrag aus */etc/fstab*:

```
# Volume              Mountpunkt        Typ     Optionen
news:/usr/spool/news  /usr/spool/news   nfs    timeo=14,intr
```

Dieses Volume kann mit dem folgenden Befehl gemountet werden:

```
# mount news:/usr/spool/news
```

Fehlt ein solcher *fstab*-Eintrag, sehen NFS-*mount*s wesentlich unangenehmer aus. Nehmen wir zum Beispiel an, daß Sie die Home-Verzeichnisse der Benutzer von einer Maschine namens moonshot aus

mounten. Um eine Blockgröße von 4 K für Schreib/Lese-Operationen zu verwenden, benutzen Sie den folgenden Befehl:

```
# mount moonshot:/home /home -o rsize=4096,wsize=4096
```

Eine vollständige Liste aller gültigen Optionen ist in der *nfs(5)*-Manpage beschrieben, die bei Rick Sladkeys NFS-fähigem *mount*-Tool enthalten ist. (Sie finden es in Rik Faiths *util-linux*-Paket.) Nachfolgend ein Teil der Optionen, die Sie möglicherweise verwenden möchten:

rsize=n und wsize=n

Bestimmt die bei Schreib- bzw. Leseanforderungen von NFS-Clients verwendete Datagramm-Größe. Im Moment ist diese aufgrund der oben beschriebenen Einschränkungen in der UDP-Datagramm-Größe auf 1024 Byte voreingestellt.

timeo=n

Bestimmt die Zeit (in Zehntelsekunden), die ein NFS-Client auf den Abschluß einer Anforderung wartet. Der voreingestellte Wert ist 7 (0,7 Sekunden).

hard

Markiere dieses Volume explizit als »hart« gemountet. Per Voreinstellung aktiviert.

soft

»Weiches« Mounten des Verzeichnisses (im Gegensatz zu hartem Mounten).

intr

Unterbrechung von NFS-Aufrufen über Signale möglich. Nützlich, wenn ein Server nicht antwortet und der Client abgebrochen werden muß.

Mit Ausnahme von `rsize` und `wsize` wirken sich all diese Optionen auf das Verhalten des Client aus, wenn der Server kurzfristig nicht erreichbar sein sollte. Sie spielen auf folgende Weise zusammen: Sendet der Client eine Anforderung an den NFS-Server, erwartet er, daß die Operation innerhalb einer bestimmten Zeit abgeschlossen ist. Diese Zeit kann mit der Option `timeout` festgelegt werden. Wird innerhalb dieser Zeit keine Bestätigung empfangen, kommt es zu einem sogenannten *kleinen Timeout*. Die Operation wird nun wiederholt, wobei das Timeout-Intervall verdoppelt wird. Wird der maximale Timeout von 60 Sekunden erreicht, tritt ein *großer Timeout* auf.

Per Voreinstellung gibt ein Client bei einem schwerwiegenden Timeout eine Fehlermeldung auf der Console aus und versucht es erneut. Dabei wird das ursprüngliche Timeout-Intervall verdoppelt. Theoretisch könnte dies immer so weitergehen. Volumes, die eine Operation störrisch wiederholen, bis der Server wieder verfügbar ist, werden als *fest gemountet* (hard-mounted) bezeichnet. Die andere Variante wird als *weich gemountet* (soft-mounted) bezeichnet und generiert einen I/O-Fehler für den rufenden Prozeß, wenn ein schwerwiegender Fehler auftritt.

Ob Sie ein Volume fest mounten oder nicht, ist teilweise von Ihrem Geschmack, teilweise aber auch von der Art der Informationen abhängig, auf die Sie über dieses Volume zugreifen wollen. Wenn Sie etwa Ihre X-Programme über NFS mounten, wollen Sie wahrscheinlich nicht, daß Ihre X-Session verrückt spielt, nur weil jemand das Netzwerk zum Erliegen gebracht hat, indem er sieben Kopien von *xv* gleichzeitig benutzt, oder weil er kurz den Ethernet-Stecker gezogen hat. Indem Sie diese Programme fest mounten, stellen Sie sicher, daß der Computer so lange wartet, bis die Verbindung mit dem NFS-Server wiederhergestellt ist. Auf der anderen Seite brauchen unkritische Daten wie

NFS-gemountete News-Partitionen oder FTP-Archive nicht fest gemountet zu sein. Ist die entfernte Maschine zeitweise nicht erreichbar oder heruntergefahren, hängt sich Ihre Session nicht auf. Ist Ihre Netzwerk-Verbindung etwas instabil, oder läuft sie über einen überlasteten Router, können Sie den anfänglichen Timeout mit Hilfe der Option `timeo` erhöhen oder die Volumes fest mounten. Dabei sollten Sie es aber ermöglichen, daß Signale die NFS-Aufrufe unterbrechen dürfen, damit Sie einen hängenden Dateizugriff unterbrechen können.

Der *mountd*-Dämon hält normalerweiße auf die eine oder andere Art fest, welche Verzeichnisse von welchen Hosts gemountet wurden. Diese Information kann mit dem Programm *showmount* ausgegeben werden, die ebenfalls Teil des NFS-Serverpakets ist. Linux-*mountd* dagegen kennt diese Möglichkeit noch nicht.

Die NFS-Dämonen

Wenn Sie anderen Hosts NFS-Dienste anbieten wollen, müssen die *nfsd*- und *mountd*-Dämonen auf Ihrer Maschine laufen. Als RPC-basierte Programme werden sie nicht von *inetd* verwaltet, sondern werden während der Bootphase gestartet und registrieren sich selbst beim Portmapper. Daher müssen Sie sicherstellen, daß die Programme erst gestartet werden, wenn *rpc.portmap* schon läuft. Normalerweise fügen Sie die folgenden Zeilen in Ihr *rc.inet2*-Script ein:

```
if [ -x /usr/sbin/rpc.mountd ]; then
        /usr/sbin/rpc.mountd; echo -n " mountd"
fi
if [ -x /usr/sbin/rpc.nfsd ]; then
        /usr/sbin/rpc.nfsd; echo -n " nfsd"
fi
```

Die Besitzinformationen über Dateien, die ein NFS-Dämon an seine Clients liefert, bestehen üblicherweise nur aus den numerischen Benutzer- und Gruppen-IDs. Wenn der Client und der Server dieselben Benutzer- und Gruppennamen mit diesen numerischen IDs verbinden, spricht man von einem gemeinsamen UID/GID-Raum. Dies ist beispielsweise der Fall, wenn Sie NIS einsetzen, um die *passwd*-Informationen an alle Hosts in Ihrem LAN weiterzugeben.

In manchen Fällen stimmen sie aber nicht überein. Statt nun die UIDs und GIDs der Clients mit denen auf dem Server abzugleichen, können Sie auf dem Client den *ugidd*-Dämon einsetzen, um dieses Problem zu umgehen. Mit der nachfolgend erläuterten Option `map_daemon` können Sie *nfsd* anweisen, mit Hilfe von *ugidd* den UID/GID-Raum des Servers auf den UID/GID-Raum des Client abzubilden.

ugidd ist ein RPC-basierter Server, der, ebenso wie *nfsd* und *mountd*, aus *rc.inet2* heraus gestartet wird.

```
if [ -x /usr/sbin/rpc.ugidd ]; then
        /usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

Die exports-Datei

Für jeden Client werden die Zugriffsmöglichkeiten bestimmt, über die auf die Dateien auf dem Server zugegriffen werden kann. Dieser Zugriff wird in der Datei */etc/exports* festgelegt, die die gemeinsam genutzten Dateien enthält.

Per Voreinstellung erlaubt *mount* niemandem, Verzeichnisse des lokalen Hosts über NFS zu mounten, was eine sehr vernünftige Einstellung ist. Um einem oder mehreren Hosts den NFS-Zugriff auf ein Verzeichnis zu erlauben, müssen Sie es *exportieren*, d. h. in der Datei *export* eintragen. Eine Beispieldatei könnte so aussehen:

```
# exports-Datei für vlager
/home             vale(rw) vstout(rw) vlight(rw)
/usr/X386         vale(ro) vstout(ro) vlight(ro)
/usr/TeX          vale(ro) vstout(ro) vlight(ro)
/                 vale(rw,no_root_squash)
/home/ftp         (ro)
```

Jede Zeile definiert ein Verzeichnis und die Hosts, die es mounten dürfen. Ein Hostname ist üblicherweise ein voll qualifizierter Domainname, kann zusätzlich aber auch die Platzhalter `*` und `?` enthalten, die auf dieselbe Weise funktionieren wie bei der Bourne-Shell. Beispielsweise wird bei `lab*.foo.com` sowohl *lab01.foo.com* als auch *laber.foo.com* akzeptiert. Wird wie im obigen Beispiel beim */home/ftp*-Verzeichnis kein Hostname angegeben, darf jeder Host dieses Verzeichnis mounten.

Bei der Prüfung eines Client-Host gegen die exports-Datei ermittelt *mountd* den Hostnamen des Client mit Hilfe eines *gethostbyaddr*-Aufrufs. Unter DNS liefert dieser Aufruf den kanonischen Hostnamen des Client zurück, d. h. Sie dürfen keine Aliases in *exports* verwenden. Ohne DNS wird der erste Hostname aus *hosts* zurückgeliefert, bei dem die Adresse des Client übereinstimmt.

Dem Host-Namen kann eine Liste mit durch Kommata getrennten Optionen folgen, die in eckigen Klammern eingeschlossen sind. Diese Optionen können die folgenden Werte annehmen:

insecure

Auf diese Maschine ist nicht-authentisierter Zugriff erlaubt.

unix-rpc

Von dieser Maschine wird UNIX-Domain RPC-Authentisierung benötigt. Dies bedeutet einfach, daß Anforderungen von reservierten Internet-Ports (die Port-Nummer muß kleiner als 1024 sein) stammen müssen. Diese Option ist per Voreinstellung aktiviert.

secure-rpc

Von dieser Maschine wird Secure-RPC-Authentizierung benötigt. Dies ist bislang noch nicht implementiert. Siehe Suns Dokumentation zu Secure-RPC.

kerberos

Von dieser Maschine wird Kerberos-Authentizierung benötigt. Dies ist bislang noch nicht implementiert. Siehe die MIT-Dokumentation zum Kerberos-Authentizierungssystem.

root_squash

Ein Sicherheits-Feature, das den Super-Usern der spezifizierten Hosts jegliche besonderen Rechte verweigert, indem Anforderungen von UID 0 des Client auf UID 65534 (-2) auf dem Server abgebildet werden. Diese UID sollte dem Benutzer nobody zugewiesen sein. Diese Option ist per Voreinstellung aktiviert.

no_root_squash

Bildet Anforderungen von UID 0 nicht ab.

ro

Mounte die Datei-Hierarchie ohne Schreibmöglichkeit.

rw

Mounte die Datei-Hierarchie mit Schreib-/Leserechten. Diese Option ist per Voreinstellung aktiv.

link_relative

Wandle absolute symbolische Links (die mit einem Slash beginnen) in relative Links um, durch Voranstellen von `../`. Der String wird so oft vorangestellt, wie es dauert, von dem Verzeichnis, das den Link enthält, bis zum Root-Verzeichnis zu gelangen. Diese Option macht nur Sinn, wenn das gesamte Dateisystem des Host gemountet ist. Anderenfalls könnten einige Links ins Nirgendwo zeigen, oder, noch schlimmer, auf Dateien, auf die niemals gezeigt werden sollte. Diese Option ist per Voreinstellung aktiviert.

link_absolute

Lasse alle symbolischen Links unverändert (das normale Verhalten bei von Sun abgeleiteten NFS-Servern).

map_identity

Die Option `map_identity` weist den Server an, gleiche UIDs und GIDs für Clients und Server zu erwarten. Diese Option ist per Voreinstellung aktiv.

map_daemon

Diese Option weist den NFS-Server an, nicht denselben UID/GID-Raum für Clients und Server zu erwarten. *nfsd* baut dann eine Liste auf, die IDs zwischen Client und Server durch Abfrage des *ugidd*-Dämons des Client abfragt.

Fehler beim Lesen der *exports*-Datei wird an *syslogd*s `daemon`-Einrichtung auf der Ebene `notice` weitergegeben, wenn *nfsd* oder *mountd* hochfährt.

Beachten Sie, daß Hostnamen aus den IP-Adressen des Client durch »Reverse Mapping« ermittelt werden, d. h. der Resolver muß korrekt konfiguriert sein. Wenn Sie mit BIND arbeiten und sicherheitsbewußt sind, sollten Sie den Spoof-Schutz in Ihrer *host.conf* aktivieren.

Der Linux-Auto-Mounter

Manchmal wäre es Verschwendung, alle NFS-Volumes zu mounten, auf die ein Benutzer möglicherweise zugreifen möchte, sei es wegen der reinen Menge an Volumes, die gemountet werden müßten, sei es wegen der Zeit, die beim Start benötigt würde. Eine gute Alternative hierzu ist ein sogenannter *Auto-Mounter*. Dabei handelt es sich um einen Dämon, der jedes NFS-Volume automatisch und völlig transparent mountet und nach einer gewissen Zeit wieder unmountet, wenn es nicht verwendet

wurde. Einer der besonders kleveren Züge bei einem Auto-Mounter ist seine Fähigkeit, ein bestimmtes Volume von alternativen Servern zu mounten. Sie können beispielsweise Kopien Ihrer X-Programme und Support-Dateien auf zwei oder drei Hosts verteilt haben, die alle anderen Hosts über NFS mounten. Mit dem Auto-Mounter können Sie nun für diese drei angeben, daß sie auf */usr/X386* gemountet werden sollen. Der Auto-Mounter wird dann versuchen, jeden der drei zu mounten, so lange, bis ein Versuch erfolgreich war.

Der unter Linux am häufigsten verwendete Auto-Mounter ist *amd*. Er wurde ursprünglich von Jan-Simon Pendry geschrieben und von Mitch D'Souza auf Linux portiert. Die aktuelle Version ist *amd-5.3*.

Die Beschreibung von *amd* geht über den Rahmen dieses Kapitels hinaus. Für ein gutes Handbuch sei auf die Quellen verwiesen; sie enthalten eine Texinfo-Datei mit sehr detaillierten Informationen.

Fußnoten

(1)

Rick ist unter jrs@world.std.com zu erreichen.

(2)

Sie können das Argument t nfs weglassen, weil mount anhand des Doppelpunkts erkennt, daß ein NFS-Volume gemeint ist.

(3)

Das Problem mit Write-Behind besteht darin, daß der Kernel-Buffercache mit Device/Inode-Paaren indiziert wird und daher nicht für über NFS gemountete Dateisysteme verwendet werden kann.

(4)

Wie Alan Cox mir erklärte: Die NFS-Spezifikation schreibt vor, daß jeder Datenblock auf die Platte geschrieben werden muß, bevor das OK zurückgeliefert wird. Weil BSD-basierte Kernel nur in der Lage sind, an der Pagegröße (4 K) orientierte Schreiboperationen durchzuführen, führt das Schreiben von vier 1 K großen Blöcken an einen BSD-basierten NFS-Server zu vier Schreiboperationen von je 4K.

(5)

Man sagt nicht Dateisystem, weil es kein richtiges Dateisystem ist.

Kapitel 12
Verwalten von TaylorUUCP

UUCP wurde in den späten Siebzigern von Mike Lesk in den AT&T Bell Laboratories entwickelt. Es sollte ein einfaches Wähl-Netzwerk über normale Telefonleitungen zur Verfügung stellen. Weil die meisten Leute, die E-Mail und Usenet-News auf ihren Heimcomputern anwenden möchten, immer noch über Modems miteinander kommunizieren, ist UUCP bis heute recht weit verbreitet. Obwohl es eine ganze Reihe von UUCP-Implementierungen für eine Vielzahl unterschiedlicher Hardware-Plattformen und Betriebssysteme gibt, sind diese doch in hohem Maße zueinander kompatibel.

Aber wie beim größten Teil der Software, die sich über die Jahre irgendwie als »Standard« etabliert hat, gibt es kein UUCP, das man als *das* UUCP bezeichnen würde. Seit der ersten Version im Jahre 1976 war es einer ständigen Evolution unterworfen. Momentan gibt es zwei Hauptspezies, die sich hauptsächlich in der unterstützten Hardware und ihrer Konfiguration unterscheiden. Von diesen beiden Hauptversionen existieren die unterschiedlichsten Implementierungen, die jeweils leicht von den Ursprungsversionen abweichen.

Eine Spezies wird Version-2-UUCP genannt, was auf eine Implementierung von Mike Lesk, David A. Novitz und Greg Chessona aus dem Jahre 1977 zurückdatiert werden kann. Obwohl sie recht alt ist, wird sie noch häufig eingesetzt. Neuere Implementierungen von Version 2 bieten heute viel von dem Komfort der neueren UUCP-Arten.

Die zweite Sorte wurde 1983 entwickelt und wird allgemein als BNU (Basic Networking Utilities), HoneyDanBer-UUCP oder kurz HDB bezeichnet. Der Name wurde aus den Namen der Autoren -- P. Honeyman, D. A. Novitz und B. E. Redman -- abgeleitet. HDB wurde entwickelt, um einige Defizite der UUCP-Version 2 zu beseitigen. So wurden etwa neue Transfer-Protokolle aufgenommen, und das Spool-Verzeichnis ist so aufgeteilt worden, daß es nun ein Verzeichnis für jede Site gibt, mit der Sie Daten austauschen.

Die Implementierung von UUCP, die momentan mit Linux geliefert wird, ist Taylor-UUCP 1.05.(1) Neben den traditionellen Konfigurations-Dateien kann Taylor-UUCP auch so kompiliert werden, daß es die neuen, d. h. Taylor-Konfigurations-Dateien, interpretiert.

Version 1.06 wurde kürzlich veröffentlicht und wird in Kürze ihren Weg in die meisten Distributionen finden. Die Unterschiede zwischen diesen Versionen betreffen meist Features, die Sie niemals benutzen werden. Sie sollten also in der Lage sein, Taylor-UUCP 1.06 mit den Informationen aus diesem Buch zu konfigurieren.

So wie es in den meisten Linux-Distributionen enthalten ist, ist Taylor-UUCP üblicherweise für BNU-Kompatibilität oder für das Taylor-Konfigurations-Schema (oder beides) kompiliert. Das Taylor-Schema ist wesentlich flexibler und wohl auch einfacher zu verstehen als die häufig doch recht obskuren BNU-Konfigurations-Dateien. Ich werde also nachfolgend das Taylor-Schema behandeln.

Der Sinn dieses Kapitels ist nicht, Ihnen eine ausführliche Beschreibung jeder Kommandozeilen-Option der

verschiedenen UUCP-Befehle zu geben. Vielmehr soll gezeigt werden, wie Sie einen funktionierenden UUCP-Knoten aufbauen können. Der erste Abschnitt gibt eine einfache Einführung, wie UUCP entfernte Programmausführung (remote execution) und Datei-Transfers implementiert. Wenn Ihnen UUCP nicht ganz fremd ist, können Sie zum Abschnitt *UUCP-Konfigurations-Dateien* springen, der die verschiedenen Dateien beschreibt, mit denen UUCP eingerichtet wird.

Allerdings setzen wir voraus, daß Ihnen die Benutzerprogramme des UUCP-Pakets bekannt sind. Dies sind *uucp* und *uux*. Eine Beschreibung können Sie den Online-Manpages entnehmen.

Neben den allgemein zugänglichen Programmen *uux* und *uucp* enthält das UUCP-Paket eine Reihe von Befehlen, die nur administrativen Zwecken dienen. Diese werden dazu genutzt, den UUCP-Transport über Ihren Knoten zu überwachen, alte Log-Dateien zu entfernen oder Statistiken zu erstellen. Keines dieser Programme wird hier beschrieben, weil sie bei den Hauptaufgaben von UUCP nur eine untergeordnete Rolle spielen. Nebenbei gesagt, sind sie sehr gut dokumentiert und recht einfach zu verstehen. Daneben gibt es noch eine dritte Kategorie, die die eigentlichen »Arbeitspferde« für UUCP stellt: *uucico* (wobei *cico* für copy-in copy-out steht) und *uuxqt*, das von einem entfernten Host gesandte Jobs ausführt. Auch diese werden in diesem Kapitel behandelt.

Diejenigen unter Ihnen, die in diesem Kapitel nicht alles finden, was sie benötigen, sollten die mit dem UUCP-Paket gelieferte Dokumentation lesen. Dies ist eine Reihe von Texinfo-Dateien, die die Einrichtung bei Verwendung des Taylor-Konfigurations-Schemas beschreiben.

Wenn Sie BNU- oder sogar (Oh Schreck!) Version-2-Konfigurations-Dateien verwenden wollen, kann ich das Buch *Managing UUCP and Usenet* von O'Reilly & Associates empfehlen. Ich fand es sehr hilfreich. Eine andere sehr gute Informationsquelle zu UUCP unter Linux ist Vince Skahans UUCP-HOWTO, das regelmäßig in *comp.os.linux.announce* gepostet wird.

Es existiert auch eine Newsgruppe zum Thema UUCP namens *comp.mail.uucp*. Wenn Sie Fragen zu Taylor-UUCP haben, sollten Sie sie lieber dort stellen als in den *comp.os.linux*-Gruppen.

UUCP-Transfer und entfernte Programmausführung

Das Konzept der *Jobs* ist für das Verständnis von UUCP von elementarer Bedeutung. Jeder Datentransfer, den ein Benutzer mit *uucp* oder *uux* initiiert, wird Job genannt. Er besteht aus einem Befehl, der auf einem entfernten System ausgeführt wird, und einer Reihe von Dateien, die zwischen Sites übertragen werden sollen. Einer dieser beiden Teile kann dabei jeweils fehlen.

Nehmen wir zum Beispiel an, Sie haben den folgenden Befehl auf Ihrem Host eingegeben. Dieser sorgt dafür, daß UUCP die Datei *netguide.ps* auf den Host **pablo** kopiert und dort den Befehl *lpr* ausführt, mit dem die Datei gedruckt wird.

```
$ uux -r pablo!lpr !netguide.ps
```

UUCP ruft üblicherweise das entfernte System nicht direkt an, um einen Job auszuführen (sonst könnten Sie das auch mit *kermit* erledigen). Statt dessen wird der Job vorübergehend zwischengespeichert. Diese Technik wird *Spooling* genannt. Der Verzeichnisbaum, unter dem die Jobs gespeichert werden, wird daher auch *Spool-Verzeichnis* genannt und steht üblicherweise unter */var/spool/uucp*. In unserem Beispiel würde die Job-Beschreibung Informationen zu dem Befehl (*lpr*) enthalten, der ausgeführt werden soll, zum Benutzer, der die Ausführung angefordert hat, sowie zu einer Reihe weiterer Dinge. Neben der Job-Beschreibung muß UUCP auch die Eingabedatei *netguide.ps* speichern.

Der exakte Ort und der Name von Spool-Dateien können, abhängig von einigen Optionen während des Kompilierens, variieren. HDB-kompatible UUCPs speichern Spool-Dateien üblicherweise in einem */var/spool/uucp*-Unterverzeichnis mit dem Namen der entfernten Site. Wurde es für die Taylor-Konfiguration kompiliert, erzeugt UUCP Unterverzeichnisse unter Site-spezifischen Spool-Verzeichnissen für verschiedene Arten von Spool-Dateien.

In regelmäßigen Intervallen wählt UUCP das entfernte System an. Steht die Verbindung zum entfernten System, überträgt UUCP die den Job beschreibenden Dateien sowie alle Eingabedateien. Die eingehenden Jobs werden nicht sofort ausgeführt, sondern erst nachdem die Verbindung unterbrochen wurde. Dies wird im allgemeinen durch *uuxqt* erledigt, das auch dafür sorgt, daß alle Jobs weitergeleitet werden (»Forwarding«), die für eine andere Site bestimmt sind.

Um zwischen wichtigen und weniger wichtigen Jobs zu unterscheiden, verknüpft UUCP eine Klasse (*Grade*) mit jedem Job. Dabei handelt es sich (bei absteigender Wichtigkeit) um einen einzelnen Buchstaben von 0 bis 9, A bis Z und a bis z. Mail wird normalerweise mit der Klasse B oder C gespoolt, während News mit N gespoolt werden. Jobs mit einer höheren Klasse werden früher übertragen. Sie können die Spool-Klasse mit der Option *-g* vergeben, wenn Sie *uucp* bzw. *uux* starten.

Sie können auch den Transfer von Jobs zu bestimmten Zeiten unterbinden, wenn eine bestimmte Klasse unterschritten wird. Dieser Wert wird als die *maximale Spool-Klasse* bezeichnet, die während einer Übertragung erlaubt ist. Sie ist normalerweise auf z voreingestellt. Beachten Sie den terminologischen Widerspruch an dieser Stelle: Eine Datei wird nur dann übertragen, wenn sie *gleich oder größer* der maximalen Spool-Klasse ist.

Die interne Arbeitsweise von uucico

Um zu verstehen, warum *uucico* bestimmte Dinge wissen muß, ist eine kurze Beschreibung darüber hilfreich, wie es die eigentliche Verbindung zu einem entfernten System herstellt.

Wenn Sie *uucico -s* `system` von der Kommandozeile aus ausführen, muß *uucico* zuerst eine physikalische Verbindung herstellen. Die durchzuführenden Aktionen sind von der Verbindungsart abhängig -- wird eine Telefonleitung verwendet, muß ein Modem gefunden und herausgewählt werden. Unter TCP muß *gethostbyname* aufgerufen werden, um den Namen in eine Netzwerk-Adresse umzuwandeln. Der zu öffnende Port muß ermittelt und die Adresse an den entsprechenden Socket gebunden werden.

Nachdem die Verbindung aufgebaut worden ist, muß eine Autorisierungs-Prozedur durchgeführt werden. Diese Prozedur besteht normalerweise darin, daß das entfernte System nach einem Loginnamen und eventuell nach einem Paßwort fragt. Dies wird im allgemeinen als *Login-Chat* bezeichnet. Die Autorisierungs-Prozedur wird entweder vom normalen *getty*/*Login*-Paket oder, bei TCP-Sockets, von *uucico* selbst durchgeführt. Ist die Autorisierung erfolgreich verlaufen, wird am anderen Ende *uucico* gestartet. Die lokale Kopie von *uucico*, die die Verbindung initiiert hat, wird als *Master*, die entfernte Kopie als *Slave* bezeichnet.

Als nächstes folgt eine *Handshake-Phase*: Der Master sendet nun seinen Hostnamen, zusammen mit einigen Flags. Der Slave überprüft diesen Hostnamen auf Login-Erlaubnis, Senden und Empfangen von Dateien etc. Die Flags beschreiben (unter anderem) die maximale Klasse der zu übertragenden Dateien. Falls aktiviert, wird ein Kommunikationszähler, die *Call Sequence Number*, an dieser Stelle überprüft. Mit dieser Option verwalten beide Seiten jeweils die Anzahl erfolgreicher Verbindungen, die nun miteinander verglichen werden. Stimmen sie nicht überein, schlägt der Handshake fehl. Diese Option ist nützlich, um sich vor Eindringlingen zu schützen.

Zum Schluß versuchen sich beide *uucico*s auf ein gemeinsames *Übertragungs-Protokoll* zu einigen. Dieses Protokoll legt fest, wie Daten übertragen, auf ihre Konsistenz hin überprüft und im Fehlerfall erneut übertragen werden. Wegen der verschiedenen unterstützten Verbindungsarten müssen auch verschiedene Protokolle unterstützt werden. So benötigen Telefonverbindungen ein »sicheres« Protokoll, das pingelig auf Fehler achtet, während die TCP-Übertragung ausreichend zuverlässig ist und daher ein effizienteres Protokoll verwenden kann, bei dem viele der zusätzlichen Fehlerprüfungen weggelassen werden können.

Nachdem der Handshake durchgeführt wurde, beginnt die eigentliche Übertragungsphase. Beide Seiten aktivieren den vereinbarten Protokoll-Treiber. An diesem Punkt führen die Treiber eventuell eine Protokoll-spezifische Initialisierungs-Sequenz durch.

Der Master sendet dann alle aufgelaufenen Dateien, deren Spool-Klasse ausreichend hoch ist, an das entfernte

System. Ist die Übertragung beendet, wird der Slave darüber informiert, daß die Übertragung abgeschlossen ist und daß er nun aufhängen kann. Der Slave kann nun die Verbindung unterbrechen oder die Kommunikation selbst übernehmen. In diesem Fall werden also die Rollen getauscht: Das entfernte System wird zum Master und das lokale wird zum Slave. Der neue Master überträgt nun seine Dateien. Ist auch diese Übertragung abgeschlossen, tauschen beide *uucico*s Terminierungs-Nachrichten miteinander aus und schließen die Verbindung.

Wir gehen an dieser Stelle nicht detaillierter darauf ein: Sehen Sie sich dazu entweder die Quellen oder ein gutes Buch zu UUCP an. Im Netz kursiert auch ein wirklich antiquierter Artikel von David A. Novitz, der eine detaillierte Beschreibung des UUCP-Protokolls enthält.(2) Auch die Taylor-UUCP-FAQ beschreibt einige Details darüber, wie UUCP implementiert wurde. Es wird regelmäßig in *comp.mail.uucp* gepostet.

Kommandozeilen-Optionen von uucico

Dieser Abschnitt beschreibt die wichtigsten Kommandozeilen-Optionen für *uucico*.

-s system

: Ruft das angegebene `system` an, wenn dies durch Rufzeitbeschränkungen nicht unterbunden wird.

-S `system`

: `Ruft das angegebene` `system` an, wobei etwaige Bedingungen/Einschränkungen nicht beachtet werden.

-r1

: Startet *uucico* im Master-Modus. Dies ist Standard, wenn die Optionen *-s* oder *-S* verwendet werden. Steht diese Option für sich allein, versucht *uucico*, alle in *sys* eingetragenen Dateien anzurufen, solange dies durch Ruf- oder Retry-Zeitbeschränkungen nicht unterbunden wird.

-r0

: Startet *uucico* im Slave-Modus. Dies ist Standard, wenn die Optionen *-s* oder *-S* nicht verwendet werden. Im Slave-Modus wird davon ausgegangen, daß entweder die Standard-Ein-/Ausgabe mit einem seriellen Port verbunden ist, oder daß der mit der Option *-p* angegebene TCP-Port verwendet wird.

-x `typ`, -X `typ`

: `Schalte das Debugging für den angegebenen Typ an. Verschiedene Typen können in einer durch Kommata unterteilten Liste angegeben werden. Die folgenden Typen sind gültig:` `abnormal`, `chat`, `handshake`, `uucp-proto`, `proto`, `port`, `config`, `spooldir`, `execute`, `incoming` und `outgoing`. Mit `all` werden alle Typen aktiviert. Aus Kompatibilitäts-Gründen mit anderen UUCP-Implementierungen kann statt dessen auch eine Nummer angegeben werden, die das Debugging für die ersten *n* Punkte aus der obigen Liste aktiviert.

: Debugging-Nachrichten werden in der Datei *Debug* unter */var/spool/uucp* festgehalten.

UUCP-Konfigurations-Dateien

Im Gegensatz zu einfacheren Datenübertragungs-Programmen wurde UUCP so konzipiert, daß es alle Übertragungen automatisch durchführen kann. Ist es einmal korrekt eingerichtet worden, ist ein tägliches Eingreifen des Administrators nicht mehr nötig. Die für diesen automatischen Transfer benötigten Informationen werden in einer Reihe von Konfigurations-Dateien festgehalten, die im Verzeichnis */usr/lib/uucp* gehalten werden. Die meisten dieser Dateien werden nur beim Anwählen externer Hosts verwendet.

Eine sanfte Einführung in Taylor-UUCP

Die UUCP-Konfiguration als schwierig zu bezeichnen wäre eine Untertreibung. Tatsächlich ist sie eine haarige Angelegenheit, und das manchmal etwas kurz angebundene Format der Konfigurations-Dateien macht die Dinge auch nicht gerade einfacher (obwohl das Taylor-Format im Vergleich zu den älteren Formaten von HDB oder Version 2 schon wesentlich einfacher ist).

Um Ihnen ein Gefühl dafür zu vermitteln, wie diese Konfigurations-Dateien zusammenwirken, wollen wir Ihnen die wichtigsten vorstellen und einen Blick auf einige beispielhafte Einträge werfen. Wir gehen hier nicht ins Detail; genauere Beschreibungen folgen in späteren, separaten Abschnitten. Wenn Sie UUCP auf Ihrer Maschine einrichten möchten, sollten Sie am besten auf einige Beispieldateien zurückgreifen und diese schrittweise an Ihre Bedürfnisse anpassen. Sie können dazu entweder auf die nachfolgend aufgeführten oder auf die in Ihrer Linux-Distribution enthaltenen Dateien zurückgreifen.

Alle in diesem Abschnitt behandelten Dateien sind in */usr/lib/uucp* oder einem darunterliegenden Unterverzeichnis zu finden. Manche Linux-Distributionen enthalten UUCP-Binaries, die sowohl die HDB- als auch die Taylor-Konfiguration unterstützen, und verwenden verschiedene Unterverzeichnisse für den jeweiligen Satz von Konfigurations-Dateien. Sie finden üblicherweise eine *README*-Datei in */usr/lib/uucp*.

Damit UUCP ordnungsgemäß funktionieren kann, müssen die Dateien dem Benutzer **uucp** gehören. Einige enthalten Paßwörter und Telefonnummern und sollten daher die Zugriffsrechte 600 besitzen.(3)

Die zentrale UUCP-Konfigurations-Datei ist */usr/lib/uucp/config*. Sie enthält die allgemeinen Parameter. Der wichtigste (und vorerst einzige) ist der UUCP-Name Ihres Host. In der virtuellen Brauerei wird **vstout** als UUCP-Gateway benutzt:

```
# /usr/lib/uucp/config -- UUCP-Haupt-Konfigurationsdatei
hostname         vstout
```

Die nächste wichtige Konfigurations-Datei ist *sys*. Sie enthält alle systemspezifischen Informationen zu Sites, mit denen Sie verbunden sind. Sie enthält den Namen der Site und Informationen zum Link selbst, beispielsweise die Telefonnummer bei einem Modem-Link. Ein typischer Eintrag für eine über Modem verbundene Site namens **pablo** würde wie folgt aussehen:

```
# /usr/lib/uucp/sys -- benenne UUCP-Nachbarn
# System: pablo
system          pablo
time            Any
phone           123-456
port            serial1
speed           38400
chat            ogin: vstout ssword: lorca
```

`port` benennt die zu verwendende Schnittstelle und `time` gibt die Zeit an, in der das System angerufen werden kann. `chat` gibt das Login-Chat-Skript an -- die Sequenz von Strings, die ausgetauscht werden müssen, damit *uucico* sich bei **pablo** einloggen kann. Wir kommen später noch auf Chat-Skripten zurück. Der `port`-Befehl zeigt einfach auf einen Eintrag in der *port*-Datei. (Siehe Abbildung 12--1.) Sie können jeden beliebigen Namen verwenden, solange er auf einen gültigen Eintrag in *port* verweist.

Die *port*-Datei enthält die Link-spezifischen Informationen. Bei Modem-Links enthält sie die zu verwendende Gerätedatei, die unterstützten Geschwindigkeiten und die an der Schnittstelle hängende Wähleinrichtung. Der nachfolgende Eintrag beschreibt */dev/cua1* (also COM 2), woran ein NakWell-Modem angeschlossen ist, das mit einer Geschwindigkeit von bis zu 38400 bps betrieben werden kann. Der Name des Ports ist so gewählt, daß er mit dem Portnamen aus *sys* übereinstimmt.

```
# /usr/lib/uucp/port -- UUCP-Ports
# /dev/cua1 (COM 2)
port                serial1
type                modem
device              /dev/cua1
speed               38400
dialer              nakwell
```

Die Information zur Wähleinrichtung (Dialer) wird in einer weiteren Datei namens -- Sie ahnen es -- *dial* festgehalten. Für jeden Wähleinrichtungs-Typ enthält sie grundsätzlich eine Reihe von Befehlen, die ausgeführt werden müssen, um eine Gegenseite anzurufen, deren Telefonnummer angegeben wurde. Auch dies ist ein Chat-Skript. Der Eintrag für das NakWell-Modem könnte etwa so aussehen:

```
# /usr/lib/uucp/dial -- Wähleinrichtungs-Informationen (je Wähleinrichtung)
# NakWell-Modems
dialer              nakwell
chat                "" ATZ OK ATDT\T CONNECT
```

Die mit `chat` beginnende Zeile bestimmt den Modem-Chat, d. h. die Sequenz von Befehlen, die ans Modem geschickt und vom Modem empfangen wird, um es zu initialisieren und die gewünschte Nummer zu wählen. Die Sequenz `\T` wird dabei von *uucico* durch die Telefonnummer ersetzt.

Um eine grobe Vorstellung davon zu bekommen, wie *uucico* mit diesen Konfigurations-Dateien umgeht, stellen Sie sich vor, Sie hätten den folgenden Befehl eingegeben:

```
$ uucico -s pablo
```

Zuallererst sucht *uucico* in der Datei *sys* nach **pablo**. Aus dem *sys*-Eintrag für **pablo** ersieht es, daß es den Port `serial1` verwenden soll, um die Verbindung aufzubauen. Aus der Datei *port* erfährt es, daß dies ein Modem-Port ist, an den ein NakWell-Modem angeschlossen ist.

uucico durchsucht nun *dial* nach einem Eintrag, der das NakWell-Modem beschreibt. Nachdem dieser Eintrag gefunden wurde, wird aufgrund dieser Information die serielle Schnittstelle */dev/cua1* geöffnet und der Dialer-Chat ausgeführt: Es wird *ATZ* gesendet, auf die Antwort *OK* gewartet etc. Wird der String `\T` erkannt, wird dieser durch die Telefonnummer (123-456) ersetzt, die in der *sys*-Datei angegeben wurde.

Nachdem das Modem *CONNECT* zurückliefert, steht die Verbindung, und der Modem-Chat ist komplett. *uucico* kehrt nun zur *sys*-Datei zurück und führt den Login-Chat durch. In unserem Beispiel würde es zuerst auf den Prompt *login:* warten und den Loginnamen (*vstout*) senden. Danach würde es auf den Prompt *password:* warten und das Paßwort *lorca* übertragen.

Nachdem die Autorisierung abgeschlossen ist, wird erwartet, daß die Gegenseite ihr eigenes *uucico* startet. Danach beginnen beide Seiten mit der vorher beschriebenen Handshake-Phase.

Abbildung 12--1 zeigt, auf welche Weise die Konfigurations-Dateien voneinander abhängig sind.

Abbildung 12-1. Zusammenspiel der Konfigurations-Dateien bei Taylor-UUCP

Was UUCP wissen muß

Bevor Sie damit beginnen, die UUCP-Konfigurations-Dateien zu schreiben, müssen Sie zuerst einige Informationen sammeln, die UUCP benötigt.

Zuerst müssen Sie wissen, an welche serielle Schnittstelle Ihr Modem angeschlossen ist. Normalerweise sind die (DOS-)Schnittstellen COM 1 bis COM 4 auf die Gerätedateien */dev/cua0* bis */dev/cua3* abgebildet. Die meisten Distributionen wie z. B. Slackware erzeugen einen Link namens */dev/modem*, der auf die entsprechende *cua**-Gerätedatei zeigt, und konfigurieren *kermit*, *seyon* etc. so, daß diese Datei benutzt wird. Wenn dies der Fall ist, sollten Sie */dev/modem* auch in Ihrer UUCP-Konfiguration verwenden.

Der Grund für den symbolischen Link liegt darin, daß alle Programme die serielle Schnittstelle durch eine sogenannte *Lock-Datei*sperren, um zu signalisieren, daß sie benutzt wird. Die Namen dieser Lock-Dateien setzen sich aus dem String *LCK..* und einem Gerätenamen zusammen, beispielsweise *LCK..cua1*. Benutzen Programme nun unterschiedliche Namen für dasselbe Gerät, werden die entsprechenden Lock-Dateien nicht erkannt. Die Folge wäre die Unterbrechung beider Sessions, wenn sie gemeinsam gestartet worden wären. Dies ist nicht unbedingt ungewöhnlich, wenn Sie Ihre UUCP-Aufrufe durch *crontab*-Einträge erledigen.

Details zur Einrichtung Ihrer seriellen Schnittstellen entnehmen Sie bitte dem Kapitel 4, *Konfiguration der seriellen Hardware*.

Als nächstes müssen Sie herausfinden, mit welcher Geschwindigkeit Ihr Modem und Linux kommunizieren. Diese Geschwindigkeit sollten Sie auf die höchstmögliche effektive Übertragungsrate einstellen. Die effektive Übertragungsrate kann dabei wesentlich höher sein als die eigentliche physikalische Übertragungsrate Ihres Modems. Beispielsweise senden und empfangen viele Modems Daten mit 2400 Bps. Durch Verwendung von Kompressions-Protokollen wie V.42bis kann die wirkliche Übertragungsrate auf bis zu 9600 Bps erhöht werden.

Soll UUCP irgend etwas Sinnvolles anstellen, müssen Sie natürlich auch die Telefonnummer wissen, die angerufen werden soll. Außerdem benötigen Sie einen gültigen Login-ID und eventuell ein Paßwort für diese Maschine.(4)

Sie müssen auch *ganz genau* wissen, wie Sie sich in das System einloggen. Müssen Sie die Return-Taste drücken, bevor der Login-Prompt erscheint? Wird `login:` oder `user:` ausgegeben? Dies wird für die Erstellung des *Chat-Skripts* benötigt. Wenn Sie das nicht wissen, oder wenn der normale Chat fehlschlägt, wählen Sie das System mit einem Terminal-Programm wie *kermit* oder *minicom* an, und schreiben Sie genau auf, was Sie tun müssen.

Site-Namen

Genau wie bei TCP/IP-basierten Netzwerken muß Ihr Host auch in UUCP-Netzen einen Namen besitzen. Solange Sie UUCP einfach zur Datenübertragung zu und von Sites verwenden wollen, die Sie direkt anwählen, oder die in einem lokalen Netz präsent sind, muß dieser Name keinen Standards entsprechen.(5)

Wenn Sie UUCP dagegen für einen Mail- oder News-Link verwenden, sollten Sie darüber nachdenken, den Namen über das UUCP Mapping Project registrieren zu lassen. Das UUCP Mapping Project wird im Kapitel 13, *Elektronische Post* besprochen. Selbst wenn Sie an einer Domain teilnehmen, sollten Sie einen offiziellen UUCP-Namen für Ihre Site besitzen.

Häufig wird der UUCP-Name so gewählt, daß er dem ersten Teil des voll qualifizierten Domainnamens entspricht. Ist die Domain-Adresse Ihrer Site etwa swim.twobirds.com, könnte der UUCP-Hostname swim lauten. Stellen Sie sich einfach vor, daß UUCP-Sites einander duzen. Natürlich können Sie auch einen UUCP-Namen verwenden, der mit dem voll qualifizierten Domainnamen nichts zu tun hat.

Stellen Sie aber sicher, daß der unqualifizierte Site-Name nicht in Mail-Adressen verwendet wird, solange er nicht als offizieller UUCP-Name registriert ist.(6) Im besten Fall landet Mail an einen nicht registrierten UUCP-Host irgendwo in einem großen schwarzen Sack. Wenn Sie einen Namen verwenden, der bereits von einem anderen Server registriert ist, wird die Post an diese Site weitergeleitet und wird deren Postmaster beträchtliche Kopfschmerzen bereiten.

Standardmäßig verwendet das UUCP-Paket den über *hostname* gesetzten Namen als den UUCP-Namen der Site. Dieser Name wird normalerweise im Skript */etc/rc.local* gesetzt. Wenn sich Ihr UUCP-Name vom gesetzten Hostnamen unterscheidet, müssen Sie die Option `hostname` in der *config*-Datei benutzen, um *uucico* den UUCP-Namen mitzuteilen. Dies wird nachfolgend beschrieben.

Taylor-Konfigurations-Dateien

Kehren wir wieder zu den Konfigurations-Dateien zurück. Taylor-UUCP erhält seine Informationen aus den folgenden Dateien:

config
: Das ist die Haupt-Konfigurationsdatei. Sie können hier den UUCP-Namen Ihrer Site eintragen.

sys
: Diese Datei beschreibt alle Ihnen bekannten Sites. Jeder Eintrag enthält den Namen, die Rufzeiten, die Rufnummer und den Gerätetyp der Site sowie Angaben zum Login.

port
: Enthält Einträge, die jeden verfügbaren Port beschreiben, zusammen mit der unterstützten Übertragungs-Geschwindigkeit und dem zu verwendenden Dialer.

dial
: Beschreibt die zum Aufbau einer Telefonverbindung verwendeten Dialer.

dialcode
: Enthält die Definitionen der symbolischen Wählkodes, falls Sie welche benutzen.

call
: Enthält den Loginnamen und das Paßwort, die beim Anruf eines Systems verwendet werden sollen. Wird selten benutzt.

passwd
: Enthält Loginnamen und Paßwörter, die Systeme beim Einloggen benutzen könnten. Diese Datei wird nur verwendet, wenn *uucico* seine eigene Paßwortprüfung durchführt.

Taylor-Konfigurations-Dateien bestehen normalerweise aus Zeilen, die aus Schlüsselwort-/Wert-Paaren bestehen. Das Doppelkreuz (#) leitet einen Kommentar ein, der bis zum Ende der Zeile geht. Um ein Doppelkreuz in eigener Bedeutung zu verwenden, müssen Sie ihm einen Backslash (\) voranstellen.

Es gibt eine ganze Reihe von Optionen, die Sie über die Konfigurations-Dateien einstellen können. Wir können an dieser Stelle nicht alle Parameter besprechen, sondern wollen uns auf die wichtigsten beschränken. Danach sollten Sie in der Lage sein, einen Modem-basierten UUCP-Link zu konfigurieren. Weitere Abschnitte beschreiben dann die Modifikationen, die notwendig sind, wenn Sie UUCP über TCP/IP oder eine direkte serielle Leitung verwenden wollen. Eine komplette Referenz finden Sie in den den Taylor-UUCP-Sourcen beiliegenden Texinfo-Dokumenten.

Wenn Sie glauben, daß Sie Ihr UUCP-System vollständig konfiguriert haben, können Sie Ihre Konfiguration mit dem *uuchk*-Tool (zu finden in */usr/lib/uucp*) überprüfen. *uuchk* liest Ihre Konfigurations-Dateien und gibt eine detaillierte Liste mit den Konfigurationswerten für jedes System aus.

Allgemeine Konfigurations-Optionen -- die config-Datei

Für viel mehr als den UUCP-Hostnamen werden Sie diese Datei nicht benutzen. Standardmäßig benutzt UUCP den mit *hostname* besetzten Namen, es ist aber grundsätzlich eine gute Idee, den UUCP-Namen explizit hier einzutragen. Nachfolgend eine *config*-Beispieldatei:

```
# /usr/lib/uucp/config -- UUCP-Haupt-Konfigurationsdatei
```

```
hostname        vstout
```

Eine Reihe weiterer Parameter kann hier stehen, z. B. der Name des Spool-Verzeichnisses oder die Zugriffsrechte für »anonymous UUCP«. Letztere werden noch in einem späteren Abschnitt beschrieben.

UUCP über andere Systeme informieren -- die sys-Datei

Die *sys*-Datei beschreibt die Systeme, die Ihre Maschine kennt. Ein Eintrag beginnt mit dem Schlüsselwort `system` und umfaßt alle Zeilen bis zur nächsten `system`-Direktive. Diese Zeilen beschreiben alle für diese Site spezifischen Parameter. Üblicherweise enthält ein Systemeintrag Parameter wie die Telefonnummer und den Login-Chat.

Parameter, die vor der ersten `system`-Zeile stehen, setzen Werte, die für alle Systeme gelten. Normalerweise enthält dieser Abschnitt solche Dinge wie allgemeine Protokoll-Parameter.

Nachfolgend werden die wichtigsten Felder etwas detaillierter besprochen.

Systemname

Der Befehl `system` benennt das entfernte (remote) System. Sie müssen den richtigen Namen des Systems angeben und keinen erfundenen Alias, weil *uucico* diesen Namen mit dem vom entfernten System gelieferten Namen während des Logins vergleicht.(7)

Jeder Systemname darf nur einmal vorkommen. Wenn Sie mehrere Konfigurationen für dasselbe System benutzen wollen (beispielsweise verschiedene Telefonnummern, die *uucico* nacheinander versuchen soll), können Sie »Alternativen« (*alternates*) beschreiben. Alternativen werden nachfolgend besprochen.

Telefonnummer

Wird das entfernte System über eine Telefonleitung erreicht, enthält das Feld `phone` die vom Modem zu wählende Nummer. Sie kann verschiedene Platzhalter (Tokens) enthalten, die von *uucico* während der Wählprozedur interpretiert werden. Ein Gleichheitszeichen sorgt dafür, daß auf einen sekundären Wählton gewartet wird; ein Bindestrich sorgt für eine Pause von einer Sekunde. So kommen etwa einige Telefonanlagen ins Stocken, wenn zwischen einem speziellen Zugriffs-Kode und der eigentlichen Telefonnummer nicht eine kurze Pause besteht.(8)

Jede eingebettete alphabetische Zeichenkette kann verwendet werden, um Site-abhängige Informationen wie Vorwahlen zu verstecken. Ein solcher String wird in eine Telefonnummer übersetzt, indem er in der Datei *dialcode* nachgesehen wird. Nehmen wir einfach die folgende *dialcode*-Datei an:

```
# /usr/lib/uucp/dialcode -- Übersetzung von Vorwahlnummern
Bogoham         024881
Coxton          035119
```

Mit solchen Eintragungen können Sie Telefonnummern wie `Bogoham7732` in der *sys*-Datei verwenden, was die ganze Sache vielleicht ein wenig leserlicher macht.

port und speed

Die Optionen `port` und `speed` werden benutzt, um zu bestimmen, welches Gerät für die Verbindung zum anderen System verwendet werden soll, und mit welcher Geschwindigkeit die Daten über dieses Gerät übertragen werden sollen.(9) Ein `system`-Eintrag kann jeweils eine der beiden Optionen oder beide zusammen enthalten. Wenn in *port* nach einem passenden Eintrag gesucht wird, kommen nur die Ports in Frage, die den passenden Portnamen und/oder die passende Geschwindigkeit haben.

Normalerweise sollte die Option `speed` ausreichen. Wenn Sie nur ein serielles Gerät in *port* definiert haben, wird *uucico* sowieso immer die richtige Wahl treffen, d. h. es genügt, wenn Sie die gewünschte Geschwindigkeit angeben.

Selbst wenn Sie mehrere Modems an Ihrem System hängen haben, werden Sie häufig den Ports keinen Namen zuweisen. Wenn nämlich *uucico* mehrere passende Einträge findet, versucht es jedes einzelne Gerät zu öffnen, bis es ein unbenutztes gefunden hat.

Der Login-Chat

Sie haben das Login-Chat-Skript ja bereits kennengelernt, mit dem *uucico* mitgeteilt wird, wie es sich in das entfernte System einzuloggen hat. Es setzt sich aus einer Reihe von Strings zusammen, die vom lokalen *uucico*-Prozeß erwartet und übertragen werden sollen. Die Intention dabei ist, *uucico* so lange warten zu lassen, bis die entfernte Maschine den Login-Prompt sendet, und dann den Loginnamen zu übertragen. Danach wird auf den Paßwort-Prompt gewartet und das Paßwort übertragen. Die erwarteten und zu sendenden Strings werden abwechselnd angegeben. *uucico* hängt automatisch ein Wagenrücklaufzeichen (`\r`) an jeden zu sendenden String an. Ein einfaches Chat-Skript könnte etwa wie folgt aussehen:

```
login: vstout ssword: catch22
```

Ihnen wird aufgefallen sein, daß die Felder mit den erwarteten Strings nicht den ganzen Prompt enthalten. Auf diese Weise kann aber sichergestellt werden, daß das Login auch dann erfolgreich ist, wenn das entfernte System die Meldung *Login:* anstelle von *login:* ausgibt.

uucico erlaubt auch eine Art bedingter Ausführung, z. B. wenn der *getty* der entfernten Maschine zuerst zurückgesetzt werden muß, bevor es einen Prompt sendet. Zu diesem Zweck können Sie einen »Unter-Chat« an den erwarteten String anhängen, der durch einen Bindestrich abgesetzt wird. Der Unter-Chat wird nur ausgeführt, wenn der Haupt-Chat fehlschlägt, etwa durch einen Timeout-Fehler. Eine Möglichkeit, dieses Feature zu benutzen, besteht darin, einen BREAK an die entfernte Site zu schicken, wenn kein Login-Prompt erscheint. Das folgende Beispiel ist ein Allzweck-Chat-Skript, das auch funktionieren sollte, wenn zuerst die Return-Taste gedrückt werden muß, bevor das Login erscheint. Das erste leere Argument (`""`) weist UUCP an, nicht erst auf irgendetwas zu warten, sondern den nachfolgenden String direkt zu senden.

```
"" \n\r\d\r\n\c ogin:-BREAK-ogin: vstout ssword: catch22
```

Eine ganze Reihe von speziellen Strings und Escape-Zeichen können in einem Chat-Skript vorkommen. Nachfolgend ist ein Teil der Sonderzeichen aufgeführt, die in dem zu erwartenden String vorkommen können:

»«

Der leere String. Er weist *uucico* an, nicht auf einen String zu warten, sondern direkt mit dem Senden des nachfolgenden Strings zu beginnen.

\t

Tabulatorzeichen.

\r

Wagenrücklauf (Carriage Return).

\s

Leerzeichen (Space). Notwendig, um Leerzeichen in Chat-Strings einzubetten.

\n

Zeilenvorschub (Newline).

**

Backslash-Zeichen.

Bei zu sendenden Strings können Sie zusätzlich zu den obigen noch die folgenden Escape-Zeichen und -Strings verwenden:

EOT

Zeichen für das Ende der Übertragung (»End of Transmission«, *^D).*

BREAK

Break-Zeichen.

c

Unterdrückt die Übertragung eines Wagenrücklaufzeichens am Ende des Strings.

d

Verzögert das Senden um eine Sekunde.

E

Aktiviert die Überprüfung des Echos. In diesem Fall muß *uucico* darauf warten, daß alles, was es an die Gegenseite geschickt hat, auch zurückgeschickt wird, bevor es im Chat-Skript weitergeht. Hauptsächlich bei Modem-Chats sinnvoll (die wir später noch behandeln werden). Standardmäßig ist die Echoprüfung ausgeschaltet.

e

Deaktiviert die Echoprüfung.

K

Wie BREAK.

p

Pause für einen Sekundenbruchteil.

Alternativen (Alternates)

Manchmal ist es wünschenswert, für ein System mehrere Einträge zu besitzen, beispielsweise wenn das System über mehrere Telefonnummern erreicht werden kann. Bei Taylor-UUCP ist dies durch die Definition sogenannter »Alternativen« (*Alternates*) möglich.

Eine Alternative übernimmt alle Einstellungen vom Haupteintrag und gibt nur solche Werte an, die überschrieben oder dem Haupteintrag hinzugefügt werden sollen. Eine Alternative ist vom Systemeintrag durch eine Zeile mit dem Schlüsselwort `alternate` abgesetzt.

Wenn Sie zwei Telefonnummern für **pablo** benutzen wollen, können Sie den entsprechenden *sys*-Eintrag folgendermaßen verändern:

```
system       pablo
phone        123456
... Einträge wie oben...
alternate
phone        123455
```

Wird **pablo** angerufen, verwendet *uucico* zuerst die Nummer 123456. Schlägt dies fehl, versucht es die Alternative. Der alternative Eintrag behält alle Einstellungen des Haupteintrags bei und überschreibt nur die Telefonnummer.

Einschränken der Rufzeiten

Taylor-UUCP bietet eine ganze Reihe von Möglichkeiten, die Zeiten einzuschränken, in denen Anrufe zum entfernten Rechner plaziert werden können. Dies könnte notwendig sein, wenn ein Host seine Dienste zu bestimmten Zeiten einschränkt; Sie können so aber auch Zeiten ausschließen, in denen die Telefongebühren relativ hoch sind. Rufzeiteinschränkungen können aber immer überschrieben werden, indem Sie *uucico* mit den Optionen *-S* oder *-f* aufrufen.

Standardmäßig unterbindet Taylor-UUCP den Verbindungs-Aufbau zu allen Zeiten, d. h. Sie *müssen* Zeitangaben in

irgendeiner Form in Ihre *sys*-Datei aufnehmen. Wenn Zeitbeschränkungen für Sie kein Thema sind, können Sie in der *sys*-Datei die Option `time` mit dem Wert `Any` versehen.

Den einfachsten Weg, die Rufzeit einzuschränken, bietet der Eintrag `time`, dem ein aus einem Tag- und einem Zeit-Feld bestehender String folgt. Das Tag-Feld kann eine Kombination von `Mo, Tu, We, Th, Fr, Sa, Su` oder den Schlüsselwörtern `Any`, `Never` oder `Wk` (für Werktags) enthalten. Das Zeit-Feld besteht aus zwei Werten im 24-Stunden-Format, die durch einen Bindestrich voneinander getrennt sind. Sie bestimmen den Zeitraum, in dem Anrufe durchgeführt werden können. Die Zeichen werden ohne Leerzeichen miteinander kombiniert. Mehrere Zeitspezifikationen können durch Kommata zusammengruppiert werden.

```
time            MoWe0300-0730,Fr1805-2000
```

Dieses Beispiel erlaubt Anrufe Montags und Mittwochs von 3 Uhr bis 7:30 Uhr morgens und Freitags zwischen 6:05 Uhr und 10:00 Uhr abends. Überschreitet eine Zeitangabe die Mitternachtsgrenze, z. B. `Mo1830-0600`, wird dies als Montag zwischen Mitternacht und 6 Uhr morgens sowie zwischen 6:30 Nachmittags und Mitternacht interpretiert.

Die speziellen Zeit-Strings `Any` und `Never` tun genau, was Sie von ihnen erwarten: Anrufe können zu beliebigen Zeiten bzw. gar nicht getätigt werden.

Dem Befehl `time` kann optional ein zweites Argument folgen, das die Wahlwiederholung in Minuten enthält. Schlägt der Versuch, eine Verbindung aufzubauen, fehl, verhindert *uucico* für ein bestimmtes Intervall eine erneute Anwahl des entfernten Host. Legen Sie etwa ein Wiederhol-Intervall von 5 Minuten fest, verweigert *uucico* nach dem letzten Fehler den Anruf des entfernten Host für die Dauer von 5 Minuten. Standardmäßig arbeitet *uucico* mit einem Schema, bei dem sich das Wiederhol-Intervall mit jedem Fehlversuch exponentiell erhöht.

Mit dem Befehl `timegrade` können Sie die maximale Spool-Klasse für jeden Anruf festlegen. Nehmen wir beispielsweise an, Sie haben die folgenden `timegrade`-Befehle in einen `system`-Eintrag aufgenommen:

```
timegrade            N Wk1900-0700,SaSu
timegrade            C Any
```

Auf diese Weise können Jobs mit einer Spool-Klasse von C oder höher (üblicherweise hat Mail die Klasse B oder C) bei jedem Anruf übertragen werden. News (üblicherweise Klasse N) dagegen werden nur nachts und an Wochenenden übertragen.

Genau wie `time` kann auch `timegrade` ein Wiederhol-Intervall in Minuten als optionales drittes Argument enthalten.

Ein warnendes Wort ist hier angebracht. Zuerst einmal gilt die Option `timegrade` nur für das, was *Ihr* System überträgt. Die Gegenseite kann immer noch alles übertragen, was sie will. Sie können die Option `call-timegrade` verwenden, um das andere System aufzufordern, nur Jobs ab einer bestimmten Klasse zu übertragen. Es garantiert Ihnen aber niemand, daß der andere Host sich daran hält.(10)

Außerdem wird das Feld `timegrade` nicht überprüft, wenn sich ein entferntes System bei Ihnen einwählt, d. h. alle aufgelaufenen Jobs für dieses System werden übertragen. Jedoch kann das andere System Ihr *uucico* explizit auffordern, sich auf eine bestimmte Spool-Klasse zu beschränken.

Verfügbare Geräte -- die port-Datei

Die Datei *port* klärt *uucico* über die vorhandenen Ports auf. Dabei kann es sich um Modem-Ports handeln, aber auch andere Arten wie direkte Leitungen und TCP-Sockets werden unterstützt.

Genau wie die *sys*-Datei besteht auch *port* aus separaten Einträgen, die mit dem Schlüsselwort `port` beginnen, dem der Portname folgt. Dieser Name kann im `port`-Befehl der *sys*-Datei genutzt werden. Der Name muß nicht eindeutig sein. Existieren verschiedene Ports mit demselben Namen, probiert *uucico* jeden einzelnen, bis es einen findet, der

momentan nicht verwendet wird.

Dem `port`-Befehl sollte unmittelbar der `type`-Befehl folgen, der anzeigt, welche Art von Port beschrieben wird. Gültige Typen sind `modem`, `direct` für direkte Verbindungen und `tcp` für TCP-Sockets. Fehlt der `port`-Befehl, ist der Port-Typ auf Modem voreingestellt.

In diesem Abschnitt werden nur Modem-Ports behandelt; TCP-Ports und direkte Leitungen werden in einem späteren Abschnitt besprochen.

Bei Modems und direkten Ports müssen Sie das Gerät angeben, über das »nach draußen« gewählt wird. Dazu dient der Befehl `device`. Üblicherweise handelt es sich dabei um eine spezielle Gerätedatei im */dev*-Verzeichnis, wie */dev/cua1*.(11)

Im Falle eines Modems bestimmt der Port-Eintrag auch, welche Art von Modem an den Port angeschlossen ist. Unterschiedliche Modems müssen auch verschieden konfiguriert werden. Selbst angeblich Hayes-kompatible Modems müssen nicht wirklich kompatibel untereinander sein. Darum müssen Sie *uucico* mitteilen, wie das Modem zu initialisieren ist und wie es die gewünschte Nummer wählen soll. Taylor-UUCP bewahrt die Beschreibungen aller Dialer in einer Datei namens *dial* auf. Soll eine dieser Beschreibungen verwendet werden, müssen Sie den Namen des Dialers mit dem Befehl `dialer` bekanntgeben.

Manchmal möchten Sie ein Modem auch auf verschiedene Arten einsetzen, je nachdem, welches System Sie anrufen. So verstehen es beispielsweise einige ältere Modems nicht, wenn ein Highspeed-Modem versucht, eine Verbindung mit 14400 Bps aufzubauen; sie hängen einfach auf, anstatt die Verbindung mit einer Geschwindigkeit von 9600 Bps aufzubauen. Wenn Sie wissen, daß die Site **drop** ein solch einfaches Modem benutzt, müssen Sie Ihr Modem anders einstellen, wenn Sie diesen Server anwählen. Dazu müssen Sie einen zusätzlichen Port-Eintrag in Ihrer *port*-Datei angeben, der einen anderen Dialer benutzt. Jetzt können Sie dem neuen Port einen anderen Namen wie `serial1-slow` geben und diesen bei der `port`-Direktive im **drop**-Systemeintrag in *sys* verwenden.

Eine bessere Möglichkeit besteht darin, die Ports anhand der von ihnen unterstützten Geschwindigkeiten zu unterscheiden. Die beiden Port-Einträge für die eben beschriebene Situation könnten dann wie folgt aussehen:

```
# NakWell-Modem; Highspeed-Verbindungen
port            serial1         # Portname
type            modem           # Modem-Port
device          /dev/cua1       # COM2
speed           38400           # unterstützte Geschwindigkeit
dialer          nakwell         # Dialer
# NakWell-Modem; langsame Verbindung
port            serial1         # Portname
type            modem           # Modem-Port
device          /dev/cua1       # COM2
speed           9600            # unterstützte Geschwindigkeit
dialer          nakwell-slow    # hohe Geschwindigkeit gar nicht erst versuchen
```

Der Systemeintrag für **drop** würde nun `serial1` als Portnamen enthalten, aber nur eine Geschwindigkeit von 9600 Bps erlauben. *uucico* würde dann automatisch den zweiten Port-Eintrag benutzen. Alle anderen Sites, die eine Geschwindigkeit von 38400 Bps in ihrem Systemeintrag angeben, würden weiterhin den ersten Port-Eintrag verwenden.

Wie eine Nummer gewählt wird -- die dial-Datei

Die Datei *dial* beschreibt, wie die verschiedenen Dialer verwendet werden. Traditionell spricht man bei UUCP von Dialern und nicht von Modems, weil es früher gängige Praxis war, daß eine (teure) automatische Wähleinrichtung für eine ganze Reihe von Modems genutzt wurde. Heutzutage verfügen die meisten Modems über eingebaute

Wahlunterstützung, d. h. diese Unterscheidung ist ein wenig überholt.

Dessen ungeachtet können verschiedene Dialer oder Modems auch verschiedene Konfigurationen benötigen. Diese können Sie in der Datei *dial* beschreiben. Einträge in *dial* beginnen mit dem Befehl `dialer`, der den Namen des Dialers festlegt.

Der wichtigste Eintrag neben `dialer` ist der Modem-Chat, der über den Befehl `chat` spezifiziert wird. Genau wie beim Login-Chat besteht er aus einer Reihe von Strings, die *uucico* an den Dialer sendet, und den Antworten, die es daraufhin erwartet. Dieser Befehl wird normalerweise benutzt, um das Modem in einen definierten Zustand zurückzusetzen und die gewünschte Nummer zu wählen. Der folgende `dialer`-Eintrag zeigt einen typischen Modem-Chat für ein Hayes-kompatibles Modem:

```
# NakWell-Modem; Highspeed-Verbindung
dialer          nakwell         # Dialername
chat            "" ATZ OK\r ATH1E0Q0 OK\r ATDT\T CONNECT
chat-fail       BUSY
chat-fail       ERROR
chat-fail       NO\sCARRIER
dtr-toggle      true
```

Der Modem-Chat beginnt mit `""`, dem »leeren« String. `uucico` sendet also direkt den Befehl `ATZ` an das Modem. `ATZ` ist der Hayes-Befehl zum Rücksetzen des Modems. Nun wird gewartet, bis das Modem `OK` zurückschickt, bevor der nächste Befehl übertragen wird, der das lokale Echo ausschaltet und ähnliches. Nachdem das Modem wieder mit `OK` antwortet, sendet *uucico* den Wählbefehl `ATDT`. Die Escape-Sequenz `\T` wird dabei durch die Telefonnummer ersetzt, die im Systemeintrag der *sys*-Datei gefunden wurde. *uucico* wartet dann darauf, daß das Modem die Zeichenkette `CONNECT` zurückschickt, was bedeutet, daß die Verbindung mit dem entfernten Modem erfolgreich aufgebaut werden konnte.

Natürlich passiert es auch, daß die Verbindung mit dem entfernten System nicht zustandekommt, beispielsweise, weil das andere System gerade mit jemand anderem redet und besetzt ist. In einem solchen Fall gibt das Modem eine entsprechende Fehlermeldung zurück. Modem-Chats sind nicht in der Lage, solche Meldungen zu handhaben; *uucico* wartet weiter auf den erwarteten String, bis ein Timeout erfolgt. In der UUCP-Log-Datei erscheint daher nur die nichtssagende Meldung »timed out in chat script« und nicht der wahre Grund für den Fehler.

Bei Taylor-UUCP können Sie aber *uucico* über diese Nachrichten informieren, indem Sie den Befehl `chat-fail` wie oben gezeigt verwenden. Erkennt *uucico* einen `chat-fail`-String während der Ausführung des Modem-Chats, bricht es den Anruf ab und speichert die Fehlermeldung in der UUCP-Log-Datei.

Der letzte Befehl im obigen Beispiel weist UUCP an, den Zustand der Steuerleitung DTR umzuschalten, bevor der Modem-Chat beginnt. Normalerweise aktiviert der serielle Treiber DTR (*Data Terminal Ready*), wenn der Prozeß das Gerät öffnet, um dem angeschlossenen Modem mitzuteilen, daß jemand mit ihm kommunizieren möchte. Mit `dtr-toggle` ziehen Sie DTR für kurze Zeit auf Low und dann wieder High. Viele Modems können so konfiguriert werden, daß sie beim Herunterziehen von DTR auflegen, in den Befehlsmodus gehen oder sich selbst zurücksetzen.(12)

UUCP über TCP

So absurd es auch klingen mag, so ist die Verwendung von UUCP zur Datenübertragung über TCP keine so schlechte Idee, besonders wenn große Datenmengen wie Usenet-News übertragen werden sollen. Bei TCP-basierten Links werden News im allgemeinen über das NNTP-Protokoll übertragen, bei dem Artikel individuell ohne Komprimierung und andere Optimierungen angefordert und übertragen werden. Wenn dies für große Sites mit mehreren Newsfeeds auch angemessen sein mag, so ist dies für kleinere Sites mit langsameren Verbindungen wie z. B. ISDN doch recht unvorteilhaft. Diese Sites können die Qualitäten von TCP mit dem Vorteil verknüpfen, News in großen Paketen zu

verschicken, die komprimiert, und so mit geringem Overhead übertragen werden können. Eine Standardlösung, solche Batches zu übertragen, besteht in der Verwendung von UUCP über TCP.

In *sys* würden Sie ein System, das über TCP angerufen wird, wie folgt beschreiben:

```
system          gmu
address         news.groucho.edu
time            Any
port            tcp-conn
chat            ogin: vstout word: clouseau
```

Der Befehl `address` enthält die IP-Adresse des Host oder seinen voll qualifizierten Domainnamen. Der entsprechende *port*-Eintrag liest sich wie folgt:

```
port            tcp-conn
type            tcp
service         540
```

Der Eintrag bedeutet, daß eine TCP-Verbindung verwendet werden soll, wenn ein *sys*-Eintrag `tcp-conn` benutzt. Gleichzeitig soll *uucico* versuchen, die Verbindung mit dem entfernten Host über den TCP-Netzwerk-Port 540 herzustellen. Das ist die standardmäßige Port-Nummer des UUCP-Dienstes. Anstelle der Port-Nummer können Sie auch einen symbolischen Portnamen an den `service`-Befehl übergeben. Die zu diesem Namen passende Port-Nummer wird in der Datei */etc/services* nachgesehen. Der für den UUCP-Dienst übliche Name lautet `uucpd`.

Verwendung einer direkten Verbindung

Stellen Sie sich vor, Sie verwenden eine Direktleitung von Ihrem System **vstout** zum System **tiny**. Genau wie bei einem Modem auch, müssen Sie einen entsprechenden Systemeintrag in Ihre *sys*-Datei aufnehmen. Der Befehl `port` gibt dabei den Port an, über den `tiny` angesprochen wird.

```
system          tiny
time            Any
port            direct1
speed           38400
chat            ogin: cathcart word: catch22
```

In der *port*-Datei müssen Sie den seriellen Port für diese direkte Verbindung beschreiben. Ein `dialer`-Eintrag ist nicht nötig, weil zum Wählen ja keine Veranlassung besteht.

```
port            direct1
type            direct
speed           38400
device          /dev/ttyS1
```

Wer darf was bei UUCP -- Zugriffsrechte bestimmen

Befehlsausführung

Die Aufgabe von UUCP besteht darin, Dateien von einem System auf ein anderes zu kopieren, sowie die Ausführung bestimmter Befehle auf dem entfernten Host anzufordern. Natürlich wollen Sie als Administrator kontrollieren, welche Rechte Sie anderen Systemen einräumen wollen -- die Ausführung jedes Befehls auf Ihrem System ist

definitiv keine gute Idee.

Standardmäßig erlaubt Taylor-UUCP anderen Systemen nur die Ausführung von *rmail* und *rnews* auf Ihrem Rechner. Diese Programme werden häufig für den Austausch von E-Mail und Usenet-News über UUCP verwendet. Der per Voreinstellung von *uuxqt* verwendete Suchpfad kann während der Kompilierung über eine Option bestimmt werden, sollte aber */bin*, */usr/bin* und */usr/local/bin* enthalten. Um den für ein bestimmtes System gültigen Satz von Befehlen zu ändern, können Sie das Schlüsselwort `commands` in der Datei *sys* benutzen. Auf ähnliche Weise kann der Suchpfad mit dem Befehl `command-path` angepaßt werden. Beispielsweise könnten Sie dem System **pablo** die Ausführung von *rsmtp* zusätzlich zu *rmail* und *rnews* erlauben wollen:(13)

```
system          pablo
...
commands        rmail rnews rsmtp
```

Dateitransfer

Taylor-UUCP erlaubt es Ihnen auch, den Dateitransfer bis ins kleinste Detail zu regeln. Im Extremfall können Sie den Transfer von und zu einem bestimmten System völlig unterbinden. Setzen Sie einfach `request` auf `no`, und das entfernte System wird nicht mehr in der Lage sein, Dateien von Ihrem System herunterzuladen oder auf Ihr System zu übertragen. Auf dieselbe Weise können Sie verhindern, daß die Benutzer Ihres Systems Dateien von Ihrem bzw. auf Ihr System übertragen, indem Sie `transfer` auf `no` setzen. Standardmäßig dürfen die Benutzer des lokalen und des entfernten Systems Dateien up- und downloaden.

Zusätzlich können Sie konfigurieren, aus welchen und in welche Verzeichnisse kopiert werden soll. Üblicherweise werden Sie den Zugriff für andere Systeme auf eine bestimmte Verzeichnis-Hierarchie beschränken wollen, aber gleichzeitig den lokalen Benutzern erlauben, Dateien aus ihrem Home-Verzeichnis zu übertragen. Häufig dürfen die Benutzer anderer Systeme nur Dateien übertragen, die sich im öffentlichen UUCP-Verzeichnis */var/spool/uucppublic* befinden. Dies ist der traditionelle Ort, an dem Dateien öffentlich zugänglich gemacht werden, ähnlich wie bei FTP-Servern im Internet. Auf dieses Verzeichnis bezieht man sich im allgemeinen unter Verwendung des Tilde-Zeichens.

Taylor-UUCP bietet vier verschiedene Befehle zur Konfiguration der Verzeichnisse, aus denen und in die Dateien kopiert werden dürfen. Diese sind: `local-send`, das eine Liste mit Verzeichnissen enthält, aus denen ein Benutzer Dateien lesen darf; `local-receive`, das die Liste der Verzeichnisse enthält, in die ein Benutzer Dateien kopieren darf. Dazu kommen `remote-send` und `remote-receive`, die dieselbe Aufgabe für Anforderungen von einem entfernten System übernehmen. Sehen wir uns das folgende Beispiel an:

```
system          pablo
...
local-send      /home ~
local-receive   /home ~/receive
remote-send     ~ !~/incoming !~/receive
remote-receive  ~/incoming
```

Der Eintrag `local-send` erlaubt es den Benutzern ihres Host, Dateien unter */home* und aus dem öffentlichen UUCP-Verzeichnis an **pablo** zu übertragen. Der Befehl `local-receive` erlaubt es den Benutzern, Dateien im allgemein schreibbaren Verzeichnis *receive* unter *uucppublic* oder in jedem allgemein schreibbaren Verzeichnis unter */home* zu empfangen. Der Befehl `remote-send` erlaubt es **pablo**, Dateien aus */var/spool/uucppublic* anzuforden, mit Ausnahme der Dateien aus den Verzeichnissen *incoming* und *receive*. Die Ausnahmen erkennt *uucico* an dem Ausrufezeichen vor dem Verzeichnisnamen. Die letzte Zeile erlaubt es **pablo** schließlich, alle Dateien in **incoming** abzulegen.

Das Hauptproblem beim Dateitransfer mit UUCP ist, daß es Dateien nur in Verzeichnissen ablegt, die allgemein gelesen werden können. Dies könnte Benutzer in Versuchung führen, anderen Benutzern Fallen zu stellen etc. Leider gibt es für dieses Problem keine Lösung, es sei denn, Sie verhindern den Dateitransfer unter UUCP ganz.

Weiterleitung (Forwarding)

UUCP bietet einen Mechanismus, mit dem andere Systeme Dateitransfers in Ihrem Namen erledigen. Das erlaubt Ihnen beispielsweise, daß **seci** eine Datei von **uchile** für Sie empfängt und an Ihr System sendet. Der folgende Befehl würde dies erreichen:

```
$ uucp -r seci!uchile!~/find-ls.gz ~/uchile.files.gz
```

Die Technik, einen Job über mehrere Systeme laufen zu lassen, wird *Forwarding* (Weiterleitung) genannt. Im obigen Beispiel könnte der Grund dafür, Forwarding zu benutzen, der sein, daß **seci** UUCP-Zugriff auf **uchile** besitzt, Ihr Host aber nicht. Wenn Sie ein UUCP-System betreiben, werden Sie diesen Forwarding-Service auf einige wenige Hosts beschränken wollen, bei denen Sie sicher sind, daß sie keine horrende Telefonrechnung auflaufen lassen, indem die neueste X11R6-Source-Release übertragen wurde.

Standardmäßig ist bei Taylor-UUCP das Forwarding komplett deaktiviert. Um Forwarding für ein bestimmtes System zu aktivieren, müssen sie den Befehl `forward` verwenden. Mit diesem Befehl geben Sie eine Liste von Sites an, von und zu denen das System Jobs weiterleiten kann. Der UUCP-Administrator von **seci** würde beispielsweise die folgenden Zeilen in seine *sys*-Datei aufnehmen, um **pablo** die Anforderung von Dateien von **uchile** zu ermöglichen:

```
####################
# pablo
system          pablo
...
forward         uchile
####################
# uchile
system          uchile
...
forward-to      pablo
```

Der Eintrag `forward-to` für **uchile** ist notwendig, damit die Dateien auch an **pablo** weitergeleitet werden. Anderenfalls würde UUCP sie einfach aussortieren. Dieser Eintrag ist eine Variante des `forward`-Befehls, der **uchile** nur das Senden von Dateien an **pablo** über **seci** erlaubt, aber nicht andersherum.

Um das Forwarding für jedes System zu erlauben, müssen Sie das spezielle Schlüsselwort `ANY` (in Großbuchstaben!) benutzen.

Das Einwählen in Ihr System einrichten

Wenn Sie Ihre Site so einrichten wollen, daß sich jemand einwählen kann, müssen Sie Logins auf Ihrem seriellen Port erlauben und einige Systemdateien so anpassen, daß sie UUCP-Accounts enthalten. Wie dies genau funktioniert, ist Thema dieses Abschnitts.

getty einrichten

Soll eine serielle Leitung als Einwählpunkt verwendet werden, müssen Sie einen *getty*-Prozeß für diesen Port aktivieren. Allerdings sind einige *getty*-Implementierungen für diese Aufgabe nicht richtig geeignet, weil der serielle Port häufig zum ein- und auswählen benutzt wird. Sie müssen daher sicherstellen, daß ein *getty* verwendet wird, das sich den seriellen Port mit anderen Programmen wie *uucico* oder *minicom* teilen kann. Ein solches Programm ist *uugetty* aus dem *getty_ps*-Paket. Bei den meisten Linux-Distributionen ist es dabei; suchen Sie nach *uugetty* in Ihrem */sbin*-Verzeichnis. Ein anderes Programm, das ich kenne, ist Gert Doerings *mgetty*, das auch den Empfang von Faxen unterstützt. Die neuesten Versionen dieser Programme sind auf **sunsite.unc.edu** zu finden.

Dieser Abschnitt kann nicht beschreiben, wie unterschiedlich *uugetty* und *mgetty* Logins behandeln. Ausführlichere Informationen finden Sie im Serial-HOWTO von Greg Hankins und in der zu *getty_ps* und *mgetty* gehörenden Dokumentation.

UUCP-Accounts einrichten

Als nächstes müssen Sie Benutzer-Accounts einrichten, über die sich andere Sites in Ihr System einloggen und eine UUCP-Verbindung aufbauen können. Üblicherweise vergeben Sie für jedes System einen eigenen Loginnamen. Wenn Sie einen Account für das System **pablo** einrichten, könnten Sie ihm beispielsweise den Benutzernamen **Upablo** geben.

Bei Systemen, die sich über eine serielle Schnittstelle einwählen, müssen Sie diesen Account üblicherweise in die Paßwort-Datei des Systems (*/etc/passwd*) aufnehmen. Sie sind sicher nicht schlecht beraten, wenn Sie alle UUCP-Logins in einer speziellen Gruppe wie **uuguest** aufnehmen. Das Home-Verzeichnis des Accounts sollte dabei auf das öffentliche Spool-Verzeichnis */var/spool/uucppublic* zeigen; als Login-Shell muß *uucico* verwendet werden.

Wenn Sie mit dem Shadow-Paßwort-System arbeiten, können Sie diese Arbeit mit *useradd* erledigen:

```
# useradd -d /var/spool/uucppublic -G uuguest -s /usr/lib/uucp/uucico Upablo
```

Wenn Sie die Suite nicht verwenden, müssen Sie */etc/passwd* wohl von Hand editieren und eine Zeile wie die folgende einfügen. 5000 und 150 stehen dabei für die numerischen UIDs und GIDs, die dem Benutzer **Upablo** und der Gruppe **uuguest** zugewiesen wurden.

```
Upablo:x:5000:150:UUCP Account:/var/spool/uucppublic:/usr/lib/uucp/uucico
```

Nach der Installation des Accounts müssen Sie ihn noch aktivieren, indem Sie das Paßwort mit dem Befehl *passwd* einrichten.

Um UUCP-Systeme zu bedienen, die sich über TCP mit Ihrem Server verbinden, müssen Sie *inetd* so einrichten, daß es die auf dem `uucp`-Port eingehenden Verbindungen verwaltet. Dies geschieht durch Einfügen der folgenden Zeile in Ihre */etc /inetd.conf*:(14)

```
uucp   stream  tcp   nowait  root  /usr/sbin/tcpd  /usr/lib/uucp/uucico -l
```

Durch die Option *-l* führt *uucico* seine eigene Login-Autorisierung durch. Genau wie beim normalen *login*-Programm wird auch hier nach dem Login und dem Paßwort gefragt, zur Verifizierung wird aber eine private Paßwort-Datei anstelle von */etc/passwd* benutzt. Die private Paßwort-Datei heißt */usr/lib/uucp/passwd* und enthält Paare von Loginnamen und Paßwörtern.

```
Upablo  IslaNegra
Ulorca  co'rdoba
```

Diese Datei muß natürlich **uucp** gehören und die Rechte 600 besitzen.

Wenn sich diese Datenbank für Sie so gut anhört, daß Sie sie auch für Ihre normalen seriellen Logins verwenden wollen, werden Sie wohl enttäuscht sein zu hören, daß dies ohne größere Umstände leider nicht möglich ist. Zuerst einmal benötigen Sie dazu Taylor-UUCP 1.05, weil es *getty* erlaubt, den Loginnamen des Anrufers über die Option *-u* an *uucico* weiterzugeben.(15) Dann müssen Sie *getty* dazu bringen, *uucico* anstelle der normalen */bin/login* zu starten. Mit *getty_ps* geht dies, indem Sie die Option `LOGIN` in der Konfigurations-Datei einrichten. Allerdings deaktiviert dies automatisch alle interaktiven Logins. Andererseits besitzt *mgetty* ein schönes Feature, mit dem Sie abhängig vom Benutzer-Namen unterschiedliche Login-Befehle ausgeführt werden können. So können Sie *mgetty* etwa anweisen, *uucico* für all die Benutzer zu verwenden, deren Loginname mit einem großen U beginnt, während für alle anderen Benutzer der normale *login*-Befehl Anwendung findet.

Um Ihre UUCP-Benutzer vor Anrufern zu schützen, die einen falschen Systemnamen angeben und sich deren gesamte Mail ansehen, sollten Sie `called-login`-Befehle für jeden Systemeintrag in Ihrer *sys*-Datei aufnehmen. Dies wird im nächsten Abschnitt erläutert.

Wie Sie sich vor Schwindlern schützen

Ein Hauptproblem von UUCP besteht darin, daß das anrufende System bei der Angabe seines Namens lügen kann. Es gibt seinen Namen zwar nach dem Login bekannt, aber der Server hat keine Möglichkeit, diesen zu überprüfen. Der Eindringling könnte sich also unter seinem eigenen UUCP-Account einloggen und anschließend vorgaukeln, jemand anders zu sein, um sich die Mail der anderen Site anzueignen. Dies ist besonders schwerwiegend, wenn Sie Logins via Anonymous-UUCP anbieten, wo die Paßwörter öffentlich gemacht werden.

Solange Sie sich nicht ganz sicher sind, daß alle an Ihr System angeschlossenen Sites vertrauenswürdig sind, *müssen* Sie sich gegen diese Art von Eindringlingen schützen. Das Mittel gegen diese Krankheit besteht darin, von jedem System einen bestimmten Loginnamen zu erwarten. Dieser wird durch den Befehl `called-login` in *sys* spezifiziert. Ein solcher Systemeintrag könnte etwa wie folgt aussehen:

```
system          pablo
... normale Optionen ...
called-login    Upablo
```

Das Ergebnis ist, daß jedesmal, wenn ein System sich einloggt und vorgibt, **pablo** zu sein, *uucico* prüft, ob es sich unter **Upablo** eingeloggt hat. Wenn nicht, wird das anrufende System abgewiesen und die Leitung wird unterbrochen. Sie sollten es sich zur Gewohnheit machen, den Befehl `called-login` bei jedem System zu verwenden, das Sie in Ihre *sys*-Datei aufnehmen. Es ist wichtig, dies für *alle* Systeme zu tun, gleichgültig, ob sie Ihre Site jemals anwählen oder nicht. Bei Sites, die Sie niemals anrufen, sollten Sie `called-login` auf irgendeinen wirren Namen, wie etwa neverlogsin, setzen.

Seien Sie paranoid -- Rufsequenz-Prüfungen (Call Sequence Checks)

Eine andere Möglichkeit, Eindringlinge zu erkennen und abzuwehren, bieten sogenannte Rufsequenz-Prüfungen (Call Sequence Checks). Diese helfen beim Schutz vor Eindringlingen, denen es irgendwie gelungen ist, an ein Paßwort zu gelangen und sich in das UUCP-System einzuloggen.

Bei Rufsequenz-Prüfungen speichern beide Maschinen die Anzahl der bislang aufgebauten Verbindungen. Diese Zahl wird bei jeder neuen Verbindung erhöht. Nach dem Login sendet der Anrufer seine Rufsequenz-Nummer, die vom System mit der eigenen Zahl verglichen wird. Stimmen die Zahlen nicht überein, wird die Verbindung unterbrochen. Wird die Zahl zu Beginn zufällig gewählt, haben es Angreifer schwer, die richtige Nummer zu erraten.

Aber Rufsequenz-Prüfungen tun noch mehr für Sie als das: Selbst wenn eine besonders schlaue Person Ihre Rufsequenz-Nummer und Ihr Paßwort ermittelt haben sollte, finden Sie dies heraus. Dringt ein Angreifer in Ihren

UUCP-Account ein und stiehlt Ihnen Ihre Mail, wird die Rufsequenz-Nummer um eins erhöht. Wenn *Sie* beim nächsten Mal versuchen, sich einzuloggen, wird das entfernte *uucico* Sie ablehnen, weil die Nummern nicht mehr übereinstimmen!

Haben Sie die Rufsequenz-Prüfungen aktiviert, sollten Sie Ihre Log-Dateien regelmäßig auf Fehlermeldungen überprüfen, die auf mögliche Angriffe hinweisen. Weist Ihr System die Rufsequenz-Nummer ab, die ihm von einem System angeboten wurde, schreibt *uucico* eine Nachricht wie »Out of sequence call rejected.« in die Log-Datei. Wurde Ihr System abgewiesen, weil die Sequenznummer nicht übereinstimmt, finden Sie eine Nachricht wie »Handshake failed (RBADSEQ)« in Ihrer Log-Datei vor.

Um die Rufsequenz-Prüfung zu aktivieren, müssen Sie den folgenden Befehl zu Ihrem Systemeintrag hinzufügen:

```
# Rufsequenz-Prüfung aktivieren
sequence        true
```

Außerdem müssen Sie eine Datei erzeugen, die die Sequenznummer selbst enthält. Taylor-UUCP bewahrt diese Zahl in einer Datei namens *.Sequence* im Spool-Verzeichnis des anderen Rechners auf. Sie *muß* **uucp** gehören und den Modus 600 besitzen (d. h. sie kann nur von **uucp** gelesen und geschrieben werden). Am besten initialisieren Sie diese Datei mit einer willkürlichen Zahl, auf die sich beide Seiten verständigt haben. Anderenfalls könnte ein Angreifer versuchen, die Nummer zu ermitteln, indem er beispielsweise alle Zahlen unter 60 ausprobiert.

```
# cd /var/spool/uucp/pablo
# echo 94316 > .Sequence
# chmod 600 .Sequence
# chown uucp.uucp .Sequence
```

Natürlich muß auch die Gegenseite die Rufsequenz-Prüfung aktiviert haben und mit genau derselben Nummer beginnen wie Sie.

Anonymous UUCP

Wenn Sie den Zugriff auf Ihr System über »anonymous UUCP« ermöglichen wollen, müssen Sie zuerst, wie oben beschrieben, einen speziellen Account dafür einrichten. In der Praxis wird für diesen anonymen Account häufig **uucp** als Loginname und Paßwort verwendet.

Zusätzlich müssen Sie einige Sicherheitsoptionen für unbekannte Systeme einrichten. So werden Sie üblicherweise verhindern wollen, daß diese Systeme irgendwelche Befehle auf Ihrem Rechner ausführen können. Nun können Sie diese Parameter aber nicht in der *sys*-Datei eintragen, weil der `system`-Eintrag einen Systemnamen erwartet, den Sie aber nicht haben. Taylor-UUCP löst dieses Problem mit dem Befehl `unknown`. `unknown` kann in der *config*-Datei verwendet werden, um Befehle anzugeben, die üblicherweise in einem Systemeintrag vorkommen können:

```
unknown         remote-receive ~/incoming
unknown         remote-send ~/pub
unknown         max-remote-debug none
unknown         command-path /usr/lib/uucp/anon-bin
unknown         commands rmail
```

Dies schränkt unbekannte Systeme darauf ein, Dateien unter dem Verzeichnis *pub* herunterzuladen und ins Verzeichnis *incoming* unter */var/spool/uucppublic* zu kopieren. Die nächste Zeile stellt sicher, daß *uucico* alle Anforderungen ignoriert, das lokale Debugging zu aktivieren. Die letzten beiden Zeilen ermöglichen unbekannten Systemen die Ausführung von *rmail*, wobei der von *uucico* verwendete Suchpfad allerdings auf ein privates Verzeichnis namens *anon-bin* beschränkt ist. Auf diese Weise können Sie ein spezielles *rmail* anbieten, das beispielsweise alle Mail zur Prüfung an den Super-User weiterleitet. So können anonyme Benutzer den Verwalter des Systems erreichen, ohne irgendwelche Mails an andere Sites verschicken zu können.

Um »anonymous UUCP« zu aktivieren, müssen Sie mindestens eine `unknown`-Zeile in *config* aufnehmen. Anderenfalls lehnt *uucico* alle unbekannten Systeme ab.

Low-Level-Protokolle unter UUCP

Zur Abstimmung von Session-Kontrolle und Dateitransfer mit dem anderen Ende verwendet *uucico* einen Satz von standardisierten Nachrichten. Diese werden häufig als »High-Level-Protokoll« bezeichnet. Während der Initialisierungs- und Aufhängphase werden diese einfach als einfache Zeichenketten übertragen. Während der eigentlichen Transferphase wird aber ein zusätzliches Low-Level-Protokoll genutzt, das nahezu transparent für die höheren Level ist. Auf diese Weise werden Fehlerprüfungen z. B. auf Modem-Leitungen ermöglicht.

Protokoll-Übersicht

Da UUCP über verschiedene Arten von Verbindungen wie serielle Leitungen, TCP oder sogar X.25 verwendet werden kann, werden spezifische Low-Level-Protokolle benötigt. Zusätzlich haben verschiedene UUCP-Implementierungen unterschiedliche Protokolle hervorgebracht, die ungefähr dasselbe tun.

Protokolle können in zwei Kategorien eingeteilt werden: Paket- und Stream-orientierte Protokolle. Protokolle der letztgenannten Variante übertragen Dateien als ganzes und berechnen möglichst eine Prüfsumme. Dies geschieht nahezu frei von Overhead, setzt aber eine zuverlässige Leitung voraus, weil jeder Fehler die erneute Übertragung der gesamten Datei zur Folge hat. Diese Protokolle werden üblicherweise bei TCP-Verbindungen verwendet, sind aber bei Telefonleitungen nicht angemessen. Obwohl moderne Modems eine recht gute Fehlerkorrektur besitzen, sind sie doch nicht perfekt, und es gibt keine Fehlererkennung zwischen Ihrem Computer und dem Modem.

Auf der anderen Seite teilen Paket-orientierte Protokolle die Datei in viele kleine Stücke gleicher Größe auf. Jedes Paket wird separat verschickt und empfangen, eine Prüfsumme wird berechnet, und eine Bestätigung wird an den Sender zurückgeschickt. Um dies noch effektiver zu machen, wurden sogenannte »Sliding-Window-Protokolle« eingeführt, die eine beschränkte Anzahl (ein Window) noch ausstehender Bestätigungen zu einer bestimmten Zeit ermöglichen. Das reduziert die Zeit, die *uucico* während einer Übertragung warten muß, ganz erheblich. Der im Vergleich zu den Stream-basierten Protokollen jedoch relativ große Overhead macht Paketprotokolle über TCP allerdings ineffizient.

Auch die Datenbreite macht einen Unterschied. Manchmal ist die Übertragung von 8-Bit-Zeichen über eine serielle Leitung unmöglich, weil die Verbindung über einen dummen Terminal-Server läuft, der das achte Bit abschneidet. Sollen 8-Bit-Zeichen über eine 7-Bit-Verbindung übertragen werden, müssen sie für die Übertragung besonders markiert werden. Im schlimmsten Fall wird die zu übertragende Datenmenge verdoppelt, obwohl eine von der Hardware durchgeführte Komprimierung das wieder kompensieren kann. Leitungen, bei denen beliebige 8-Bit-Zeichen übertragen werden können, werden häufig als *8-Bit-sauber* bezeichnet. Dies ist bei allen TCP-, aber auch bei den meisten Modem-Verbindungen der Fall.

Taylor-UUCP 1.04 unterstützt die folgenden Protokolle:

g

: Dies ist das am weitesten verbreitete Protokoll und sollte von nahezu allen *uucico*s verstanden werden. Es führt eine sorgfältige Fehlerprüfung durch und ist daher für rauschende Telefonverbindungen besonders geeignet. *g* benötigt eine saubere 8-Bit-Verbindung. Es ist ein Paket-orientiertes Protokoll, das die Sliding-Window-Technik benutzt.

i

: Ein bidirektionales Paket-Protokoll, das Dateien zur selben Zeit senden und empfangen kann. Es benötigt eine Vollduplex-Verbindung und einen sauberen 8-Bit-Datenpfad. Momentan wird es nur von Taylor-UUCP verstanden.

t

Dieses Protokoll ist für die Benutzung über eine TCP-Verbindung oder andere wirklich fehlerfreie Netzwerke gedacht. Es verwendet Pakete von 1024 Bytes und eine saubere 8-Bit-Verbindung.

e

Sollte grundsätzlich dasselbe tun wie *t*. Der Hauptunterschied liegt darin, daß *e* ein Stream-basiertes Protokoll ist.

f

Für die Verwendung über zuverlässige X.25-Verbindungen gedacht. Es ist ein Stream-basiertes Protokoll und erwartet einen 7 Bit breiten Datenpfad. 8-Bit-Zeichen müssen besonders gekennzeichnet werden, was es sehr ineffizient machen kann.

G

Die *System V Release 4*-Version des *g*-Protokolls. Wird auch von einigen anderen UUCP-Versionen verstanden.

a

Entspricht dem ZMODEM-Protokoll. Benötigt eine 8-Bit-Verbindung, markiert aber bestimmte Steuerzeichen wie XON und XOFF.

Einstellen des Übertragungs-Protokolls

Alle Protokolle erlauben einige Variationen bei den Paketgrößen, Timeouts und ähnlichem. Normalerweise arbeiten die voreingestellten Werte unter Standardbedingungen gut, sind aber nicht für jede Situation optimal geeignet. Zum Beispiel verwendet das *g*-Protokoll Window-Größen von 1 bis 7 und Paketgrößen in Zweierpotenzen von 64 bis 4096. (Die meisten in Linux-Distributionen enthaltenen Binaries verwenden eine Window-Größe von 7 und 128-Byte-Pakete.) Wenn Ihre Telefonleitung so verrauscht ist, daß immer mehr als 5 Prozent der Pakete verlorengehen, sollten Sie die Paketgröße verringern und das Window verkleinern. Andererseits kann der Protokoll-Overhead bei sehr guten Telefonverbindungen unnötig verschwendet werden, so daß Sie die Paketgröße auf 512 oder sogar auf 1024 Byte erhöhen können.

Taylor-UUCP stellt Ihnen mit dem `protocol-parameter`-Befehl in der *sys*-Datei einen Mechanismus zur Verfügung, mit dem Sie diese Parameter an Ihre Bedürfnisse anpassen können. Um etwa, wenn Sie mit **pablo** kommunizieren, die Paketgröße des *g*-Protokolls auf 512 zu setzen, müssen Sie folgendes hinzufügen:

```
system          pablo
...
protocol-parameter g  packet-size  512
```

Die einstellbaren Parameter und ihre Namen sind von Protokoll zu Protokoll verschieden. Eine komplette Liste können Sie der bei den Taylor-UUCP-Quellen beiliegenden Dokumentation entnehmen.

Bestimmte Protokolle wählen

Nicht jede Implementierung von *uucico* spricht und versteht jedes Protokoll. Daher müssen sich beide Prozesse während der Initialisierungs-Phase auf ein gemeinsames Protokoll verständigen. Der *uucico*-Master bietet dem Slave eine Liste unterstützter Protokolle durch Senden von *`Pprotlist`* an, aus denen sich der Slave eines aussuchen kann.

Basierend auf dem verwendeten Port-Typ (Modem, TCP oder Direkt), stellt *uucico* eine Standardliste mit Protokollen zusammen. Bei Modem- oder Direktverbindungen enthält diese Liste üblicherweise *i*, *a*, *g*, *G* und *j*. Bei TCP-Verbindungen sind *t, e, i, a, g, G, j* und *f* vorhanden. Diese Standardliste können Sie mit dem Befehl

`protocols` überschreiben, der sowohl in einem System- als auch in einem Port-Eintrag stehen kann. Sie könnten etwa den Eintrag für Ihren Modem-Port in der *port*-Datei wie folgt erweitern:

```
port            serial1
...
protocols       igG
```

Jede über diesen Port ein- oder ausgehende Verbindung müßte dann *i*, *g* oder *G* verwenden. Wird keines von diesen vom anderen System unterstützt, schlägt die Kommunikation fehl.

Fehlersuche

Dieser Abschnitt beschreibt, was bei einer UUCP-Verbindung schiefgehen kann, und gibt einige Ratschläge, wo nach Fehlern zu suchen ist. Allerdings habe ich diese Probleme aus dem Kopf zusammengestellt. Es gibt wesentlich mehr, was noch schiefgehen kann.

In jedem Fall sollten Sie mit der Option *-xall* das Debugging aktivieren und sich die Ausgaben in *Debug* im Spool-Verzeichnis ansehen. Das sollte Ihnen helfen, das Problem schnell zu lokalisieren. Ich habe es auch immer als hilfreich empfunden, den Lautsprecher des Modems zu aktivieren, wenn keine Verbindung hergestellt werden kann. Bei Hayes-kompatiblen Modems erreichen Sie dies durch Einfügen von *ATL1M1 OK* im Modem-Chat der *dial*-Datei.

Zuallererst sollten Sie immer kontrollieren, ob die Dateizugriffsrechte korrekt gesetzt sind. *uucico* sollte Setuid **uucp** sein, und alle Dateien in */usr/lib/uucp*, */var /spool/uucp* und */var/spool/uucppublic* sollten **uucp** gehören. Es gibt auch einige versteckte Dateien(16) im Spool-Verzeichnis, die auch **uucp** gehören müssen.

uucico **behauptet andauernd »Wrong time to call«:** Das bedeutet wahrscheinlich, daß Sie im Systemeintrag in *sys* keinen `time`-Eintrag spezifiziert haben, der bestimmt, wann ein System angerufen werden darf. Vielleicht haben Sie aber auch die Zeit so eingestellt, daß ein Anruf momentan einfach nicht erlaubt ist. Wird keine Zeitangabe gemacht, setzt *uucico* voraus, daß das System niemals angerufen werden kann.

uucico **beschwert sich, »site is already locked«:** Das bedeutet, daß *uucico* im Verzeichnis */var/spool/uucp* eine Lock-Datei für das andere System entdeckt hat. Die Lock-Datei könnte von einem früheren Anruf zu diesem System stammen, der abgestürzt ist oder sonstwie abrupt beendet wurde. Es ist aber auch durchaus möglich, daß ein anderer *uucico*-Prozeß irgendwo im Speicher sitzt und versucht, das entfernte System anzurufen und dabei im Chat-Skript steckengeblieben ist etc. Wenn dieser Prozeß die Verbindung nicht erfolgreich herstellen kann, brechen Sie ihn mit einem Hangup-Signal ab, und entfernen Sie alle Lock-Dateien, die von ihm übriggeblieben sind.

Ich kann die Verbindung mit der anderen Site herstellen, aber das Chat-Skript schlägt fehl: Sehen Sie sich den Text an, den Sie von der anderen Site empfangen. Wenn er unverständlich ist, kann es sich um ein Geschwindigkeitsproblem handeln. Anderenfalls sollten Sie prüfen, ob er wirklich mit dem übereinstimmt, was Ihr Chat-Skript erwartet. Denken Sie daran, daß der Chat-Skript zu Beginn auf einen String wartet. Wenn Sie den Login-Prompt empfangen und Ihren Namen senden, aber niemals den Paßwort-Prompt erhalten, sollten Sie eine Verzögerung einbauen, bevor Sie ihn senden, möglicherweise sogar zwischen den einzelnen Buchstaben. Sie könnten zu schnell für Ihr Modem sein.

Mein Modem wählt nicht: Wenn Ihr Modem nicht anzeigt, daß die DTR-Leitung aktiviert wurde, während *uucico* eine externe Verbindung aufbaut, haben Sie *uucico* möglicherweise nicht das richtige Gerät angegeben. Wenn das Modem DTR erkennt, prüfen Sie mit einem Terminal-Programm, ob Sie zum Modem schreiben können. Wenn ja, aktivieren Sie das Echo mit `\E` zu Beginn des Modem-Chats. Erfolgt kein Echo Ihrer Befehle während des Modem-Chats, sollten Sie prüfen, ob die Geschwindigkeit für Ihr Modem zu hoch oder zu niedrig ist. Läuft das Echo, sollten Sie überprüfen, ob die Modem-Antworten deaktiviert oder auf numerische Werte eingestellt sind. Prüfen Sie, ob das Chat-Skript selbst in Ordnung ist. Denken Sie daran, daß Sie zwei Backslashes schreiben müssen, um einen an das Modem zu senden.

Mein Modem versucht zu wählen, kommt aber nicht raus: Fügen Sie eine Verzögerung in die Telefonnummer ein. Das ist besonders nützlich, wenn Sie aus einer Nebenstellenanlage herauswählen. Wenn Sie normalerweise das Impuls-Wahlverfahren benutzen, probieren Sie das Mehrfrequenz-Wahlverfahren aus. In manchen Ländern haben die Telekom-Gesellschaften ihr Netz schrittweise modernisiert. Manchmal hilft die Impulswahl.

Meine Log-Datei sagt, daß ich extrem viele Pakete verliere: Das sieht nach einem Geschwindigkeitsproblem aus. Möglicherweise ist die Verbindung zwischen Computer und Modem zu langsam (denken Sie daran, die schnellste effektive Übertragungsrate einzustellen). Vielleicht ist auch Ihre Hardware zu langsam, um auf Interrupts reagieren zu können. Mit dem NSC 16550A auf Ihrer seriellen Schnittstelle sind 38 Kbps vernünftig machbar. Ohne FIFOs dagegen (z. B. der 16450-Chip) sind 9600 Bps die Grenze. Stellen Sie auch sicher, daß der Hardware-Handshake auf der seriellen Leitung aktiviert ist.

Es ist auch häufig der Fall, daß der Hardware-Handshake für den Port nicht aktiviert ist. Taylor-UUCP 1.04 besitzt keine Möglichkeit, den RTS/CTS-Handshake einzuschalten. Sie müssen ihn explizit aus der *rc.serial* mit dem folgenden Befehl aktivieren:

```
$ stty crtscts < /dev/cua3
```

Ich kann mich einloggen, aber der Handshake schlägt fehl: Nun, da kann es eine Reihe von Problemen geben. Die Ausgabe in der Log-Datei sollten Ihnen viel dazu sagen können. Achten Sie darauf, welche Protokolle der andere Rechner anbietet (er sendet den String `P`*`protlist`* während des Handshakes). Vielleicht paßt ja keins (haben Sie irgendwelche Protokolle in *sys* oder *port* gewählt?).

Sendet das andere System *RLCK*, existiert eine alte Lock-Datei für Ihren Rechner auf dem anderen System. Wenn Sie nicht bereits über eine andere Leitung mit diesem System verbunden sind, muß die Datei von der Gegenseite entfernt werden.

Wird *RBADSEQ* gesendet, wurde mit Ihnen eine Rufsequenz-Prüfung durchgeführt, aber die Zahlen stimmten nicht überein. Erhalten Sie die Meldung *RLOGIN*, wurde Ihnen das Login unter dieser ID verweigert.

Log-Dateien

Wenn Sie das UUCP-Paket so kompilieren, daß das Taylor-eigene Logging verwendet wird, gibt es nur drei globale Log-Dateien, die alle im Spool-Verzeichnis untergebracht sind. Die Haupt-Logdatei heißt *Log* und enthält alle Informationen über aufgebaute Verbindungen und übertragene Dateien. Ein typischer Ausschnitt sieht wie folgt aus (die Formatierung wurde etwas geändert, damit er auf die Seite paßt):

```
uucico pablo -- (1994-05-28 17:15:01.66 539) Calling system pablo (port cua3)
uucico pablo -- (1994-05-28 17:15:39.25 539) Login successful
uucico pablo -- (1994-05-28 17:15:39.90 539) Handshake successful
               (protocol 'g' packet size 1024 window 7)
uucico pablo postmaster (1994-05-28 17:15:43.65 539) Receiving D.pabloB04aj
uucico pablo postmaster (1994-05-28 17:15:46.51 539) Receiving X.pabloX04ai
uucico pablo postmaster (1994-05-28 17:15:48.91 539) Receiving D.pabloB04at
uucico pablo postmaster (1994-05-28 17:15:51.52 539) Receiving X.pabloX04as
uucico pablo postmaster (1994-05-28 17:15:54.01 539) Receiving D.pabloB04c2
uucico pablo postmaster (1994-05-28 17:15:57.17 539) Receiving X.pabloX04c1
uucico pablo -- (1994-05-28 17:15:59.05 539) Protocol 'g' packets: sent 15,
                resent 0, received 32
uucico pablo -- (1994-05-28 17:16:02.50 539) Call complete (26 seconds)
uuxqt pablo postmaster (1994-05-28 17:16:11.41 546) Executing X.pabloX04ai
               (rmail okir)
```

```
uuxqt pablo postmaster (1994-05-28 17:16:13.30 546) Executing X.pabloX04as
               (rmail okir)
uuxqt pablo postmaster (1994-05-28 17:16:13.51 546) Executing X.pabloX04c1
               (rmail okir)
```

Die nächste wichtige Log-Datei ist *Stats*, die die Datenübertragungs-Statistiken enthält. Der Ausschnitt aus *Stats*, der dem obigen Transfer entspricht, sieht wie folgt aus (auch hier wurden die Zeilen aufgeteilt, damit er auf die Seite paßt):

```
postmaster pablo (1994-05-28 17:15:44.78)
                  received 1714 bytes in 1.802 seconds (951 bytes/sec)
postmaster pablo (1994-05-28 17:15:46.66)
                  received 57 bytes in 0.634 seconds (89 bytes/sec)
postmaster pablo (1994-05-28 17:15:49.91)
                  received 1898 bytes in 1.599 seconds (1186 bytes/sec)
postmaster pablo (1994-05-28 17:15:51.67)
                  received 65 bytes in 0.555 seconds (117 bytes/sec)
postmaster pablo (1994-05-28 17:15:55.71)
                  received 3217 bytes in 2.254 seconds (1427 bytes/sec)
postmaster pablo (1994-05-28 17:15:57.31)
                  received 65 bytes in 0.590 seconds (110 bytes/sec)
```

Die dritte Datei ist *Debug*. An diesen Ort werden die gesamten Debugging-Informationen geschrieben. Wenn Sie das Debugging verwenden, sollten Sie sicherstellen, daß die Datei den Zugriffsmodus 600 verwendet. Abhängig vom gewählten Debug-Modus kann sie das Login und das Paßwort enthalten, die Sie verwenden, um sich in ein anderes System einzuloggen.

Die UUCP-Binaries mancher Linux-Distributionen sind so kompiliert worden, daß sie das HDB-eigene Logging verwenden. HDB-UUCP verwendet eine ganze Reihe von Log-Dateien, die alle unter */var/spool/uucp/.Log* gespeichert sind. Dieses Verzeichnis enthält drei weitere Verzeichnisse namens *uucico*, *uuxqt* und *uux*. Sie enthalten die von den jeweiligen Programmen generierten Logging-Ausgaben, die für jede Site in einer separaten Datei abgespeichert werden. Die *uucico*-Ausgabe für **pablo** wird also in *.Log/uucico/pablo* gespeichert. Entsprechend schreibt *uuxqt* seine Ausgabe in *.Log/uuxqt/pablo*. Die Zeilen, die in die verschiedenen Log-Dateien geschrieben werden, sind aber dieselben wie beim Taylor-Logging.

Aktivieren Sie die Debugging-Ausgabe beim HDB-eigenen Logging, werden die Daten im *.Admin*-Verzeichnis unter */var/spool/uucp* gespeichert. Bei ausgehenden Anrufen werden die Informationen in *.Admin/audit.local* festgehalten, während die Ausgabe von *uucico* bei eingehenden Anrufen in *.Admin/audit* gesichert wird.

Fußnoten

(1)

Geschrieben und Copyright von Ian Taylor, 1993.

(2)

Auch enthalten im System Manager's Manual von 4.4BSD.

(3)

Beachten Sie, daß, obwohl bei den meisten UUCP-Befehlen Setuid auf **uucp** gesetzt sein muß, dies beim uuchk-Programm nicht sein darf. Anderenfalls sind Benutzer in der Lage, die System-Paßwörter zu lesen, auch wenn die Dateien den Modus 600 verwenden.

(4)

Wenn Sie UUCP nur ausprobieren wollen, besorgen Sie sich die Nummer eines Archiv-Servers in Ihrer Nähe.

Notieren Sie sich Login und Paßwort diese sind öffentlich bekannt, damit Downloads möglich sind. In den meisten Fällen lauten sie uucp/uucp oder nuucp/uucp.

(5)

Die einzige Einschränkung besteht darin, daß dieser Name nicht länger als 7 Zeichen lang sein sollte, um UUCP-Implementierungen nicht zu verwirren, die auf Betriebssystemen laufen, die nur eine beschränkte Zahl von Zeichen für Dateinamen erlauben. Namen, die länger als 7 Buchstaben sind, werden von UUCP häufig abgeschnitten. Einige Versionen beschränken die Länge sogar auf 6 Zeichen.

(6)

Das UUCP Mapping Project registriert alle UUCP-Hostnamen weltweit und stellt sicher, daß sie nicht doppelt benutzt werden. Um Ihren UUCP-Namen registrieren zu lassen, wenden Sie sich an die Administratoren der Site, die Ihre Mail verwalten. Die können Ihnen weiterhelfen.

(7)

Ältere Version-2-UUCPs übertragen ihren Namen nicht, wenn sie angerufen werden. Neuere Implementierungen, darunter auch Taylor-UUCP, tun dies aber.

(8)

Zum Beispiel ist es bei den meisten Nebenstellenanlagen notwendig, eine 0 oder eine 9 vorzuwählen, um ein Amt zu bekommen.

(9)

Die Baudrate des ttys muß zumindest genauso hoch sein wie die maximale Übertragungs-Geschwindigkeit.

(10)

Läuft auf dem anderen System Taylor-UUCP, hält es sich daran.

(11)

Einige Leute verwenden statt dessen die ttyS*-Geräte, die ursprünglich nur für das Einwählen gedacht sind.

(12)

Manche Modems scheinen das allerdings nicht zu mögen und hängen sich auf.

(13)

rsmtp wird verwendet, um Mail mit »batched SMTP« auszuliefern. Dies wird in den Kapiteln zu Mail behandelt.

(14)

Beachten Sie, daß tcpd üblicherweise den Modus 700 hat, d.h. Sie müssen es als root ausführen und nicht, wie Sie es normalerweise tun würden, als **uucp**.

(15)

Die Option u ist zwar auch in Version 1.04 vorhanden, besitzt aber keine Funktion.

(16)

Das sind alle Dateien, die mit einem Punkt beginnen. Solche Dateien werden vom ls-Befehl normalerweise nicht mit ausgegeben.

Wünschen Sie mehr Informationen zu der gedruckten Version des Buches "Linux: Wegweiser durch das Netzwerk" dann klicken Sie hier.

Kapitel 13
Elektronische Post

Die Idee, persönliche Botschaften über Computernetze auszutauschen, ist so alt wie die ersten Netze selber. E-Mail (vom englischen *electronic mail*) begann als sehr primitiver Dienst, der einfach eine Datei von einer Maschine auf eine andere transportierte und an die *mailbox*-Datei der Empfängerin anhängte. Dieses Prinzip hat sich im wesentlichen bis heute nicht geändert. Allerdings haben ein stetig wachsendes Netz mit seinen komplexen Anforderungen an das Routing und die ständig zunehmende Zahl der übertragenen Botschaften ein ausgefeilteres Konzept nötig gemacht.

Heute existiert eine ganze Reihe von Standards für den Transport elektronischer Post. Internet-Sites verwenden ein Format, das ursprünglich in RFC 822 definiert, aber seitdem um mehrere Einzelheiten wie die maschinenunabhängige Übertragung von Sonderzeichen wie Umlauten ergänzt wurde. In jüngster Zeit wurde auch viel Grips auf sogenannte Multimedia-Mail verwandt. Hinter diesem Schlagwort versteckt sich die Übertragung von Bildern und Sound in Mail-Botschaften. Ein anderer, von RFC 822 unabhängiger Standard wurde von der CCITT in X.400 festgelegt.

Für Unix-Systeme steht eine ganze Reihe von Software für den Transport von E-Mail bereit. Eins der bekanntesten Programme ist das an der Universität Berkeley entwickelte *sendmail*, das auf einer ganzen Reihe von Plattformen zum Einsatz kommt. *sendmail* wurde ursprünglich von Eric Allman entwickelt, dann aber von verschiedenen Firmen und Institutionen weitergepflegt. Eine recht bekannte Version ist *sendmail-5.56c*, die auch in Kapitel 15, *Sendmail+IDA* beschrieben wird. Seit kurzem arbeitet Eric Allman wieder an einer neuen Version von *sendmail*, dem sogenannten *sendmail* V8. Der derzeit aktuelle Sproß dieser neuen Generation ist *sendmail-8.6.5*.

Der unter Linux am häufigsten benutzte Mail-Transport ist sicherlich *smail-3.1.28*, geschrieben von Curt Landon Noll und Ronald S. Karr, und findet sich in nahezu allen Linux-Distributionen. Im folgenden werden wir kurz und bündig von *smail*, Kapitel 14, *SMAIL zum Laufen bringen*, sprechen; es gibt allerdings andere (ältere) smail-Inkarnationen, mit denen Version 3.1.28 unter keinen Umständen verwechselt werden sollte. Diese alten Versionen haben mit der hier genannten Version wenig mehr als den Namen gemein.

Verglichen mit *sendmail* ist *smail* verhältnismäßig jung. Solange es darum geht, Mail für eine kleinere Site mit geringem Aufkommen und einfachem Routing zu bewältigen, unterscheiden sich beide kaum. Beim Einsatz auf größeren Sites ist *sendmail* allerdings deutlich überlegen, da seine Konfiguration wesentlich flexibler ist.

Sowohl *smail* als auch *sendmail* werden durch eine Reihe von Konfigurations-Dateien gesteuert, die an die lokalen Verhältnisse angepaßt werden müssen. Abgesehen von Informationen, die nötig sind, um das lokale Mail-System zum Fliegen zu bringen (wie beispielsweise der Rechnername), lassen sich eine ganze Reihe weiterer Parameter einstellen. *sendmail*s Konfigurations-Datei ist zu Anfang völlig unverständlich (vorsichtig ausgedrückt): Die Datei sieht in etwa so aus, als sei Ihre Katze über die Tastatur spaziert, während sie die Shift-Taste gedrückt hielten. *smail*s Konfigurations-Dateien sind dagegen klarer strukturiert und einfacher zu verstehen, lassen Ihnen dafür allerdings nicht so viele Möglichkeiten, das Verhalten des Programms an Ihre Bedürfnisse anzupassen. Trotzdem ist der Aufwand, den Sie treiben müssen, um eine kleinere UUCP- oder Internet-Site einzurichten, in ungefähr derselbe.

In diesem Kapitel werden wir uns zunächst mit den Aufgaben beschäftigen, die auf Sie als Administrator zukommen. Die Kapitel 14, *smail zum Laufen bringen* und Kapitel 15, *Sendmail+IDA* geben dann nähere Anleitungen zur Installation von *smail* und *sendmail*. Mit Hilfe dieser Kapitel sollten Sie anschließend in der Lage sein, kleinere Sites einzurichten. Das ist

natürlich noch lange nicht alles; wenn Sie wollen, können Sie Stunden um Stunden vor Ihrem Computer damit verbringen, die exotischsten Dinge zu konfigurieren. Aber so weit wollen wir hier nicht gehen.

Gegen Ende dieses Kapitels werden wir uns kurz mit der Einrichtung von *elm* beschäftigen, einem unter Unix -- und Linux --- sehr beliebten Benutzerprogramm für E-mail.

Weitere Informationen zum Thema elektronische Post finden Sie im *Electronic Mail-HOWTO* von Vince Skahan, das auf den einschlägigen FTP-Sites zu haben ist. Daneben enthalten die Quellkode-Distributionen von *elm*, *smail* und *sendmail* sehr ausführliche Dokumentationen, die die meisten Ihrer Fragen zur Konfiguration, Zucht und Pflege dieser Programme beantworten sollten. Wenn Sie nach Informationen über E-Mail im allgemeinen suchen, finden Sie in der Bibliographie am Ende des Buches eine Liste relevanter RFCs.

Was ist denn nun eine Mail?

Ein elektronischer Brief besteht zunächst aus dem Rumpf der Nachricht, also dem von Ihnen geschriebenen Text, und speziellen Daten, die die Empfänger angeben, den Absender, das Transportmedium usw., ganz ähnlich dem, was Sie sehen, wenn Sie einen normalen Briefumschlag anschauen.

Diese administrativen Daten lassen sich in zwei Kategorien unterteilen; In die erste Kategorie fallen alle Daten, deren Format vom Transportmedium abhängt, wie z. B. die Adresse des Absenders und Empfängers. Man spricht deshalb auch vom Briefumschlag, dem *envelope*. Diese Daten können von der Transport-Software transformiert werden, während sie die Botschaft weiterreicht.

Die zweite Sorte umfaßt alle Informationen, die von jedwedem Transportmechanismus unabhängig sind. Dazu gehören beispielsweise die Betreff-Zeile (engl. *subject*), die Liste aller Empfänger sowie Datum und Uhrzeit, wann die Botschaft abgeschickt wurde. In den meisten Netzen hat es sich eingebürgert, diese Informationen vor der eigentlichen Botschaft als Briefkopf anzufügen. Dieser Kopf oder *mail header* ist vom Rumpf oder *mail body* der Botschaft durch eine Leerzeile getrennt.(1)

Die meiste Mail-Software in der UNIX-Welt verwendet ein Header-Format, das in RFC 822 beschrieben wird. Sein ursprünglicher Zweck war es, einen Standard für das ARPANET zu schaffen. Dank seiner Unabhängigkeit von irgendeiner bestimmten Netzwerk-Umgebung ist RFC 822 auch von anderen Netzen übernommen worden, einschließlich vieler UUCP-basierter Netze.

RFC 822 ist allerdings nur der größte gemeinsame Nenner. In jüngerer Zeit sind verschiedene Standards entwickelt worden, die den wachsenden Bedarf an Dingen wie Verschlüsselung, Unterstützung für internationale Zeichensätze und Multimedia (MIME -- multi-media mail extensions) befriedigen sollen.

Alle diese Standards gehen davon aus, daß ein Mail-Header aus mehreren Zeilen besteht, getrennt durch Newline-Zeichen. Eine Zeile setzt sich jeweils aus einem Feldnamen zusammen, der in Spalte eins beginnt, und dem Feld selbst, das durch einen Doppelpunkt und ein Leerzeichen von Feldnamen abgesetzt ist. Das Format und die Bedeutung der einzelnen Felder variiert abhängig vom Feldnamen. Ein Feld kann in jedem Fall mehrere Zeilen fortgesetzt werden, wenn die nächste Zeile mit einem Leerzeichen (Blank oder TAB) beginnt. Die Reihenfolge der Felder spielt keine Rolle.

Ein typischer Mail-Header sieht so aus:

```
From brewhq.swb.de!ora.com!andyo Wed Apr 12 00:17:03 1995
return-Path: <brewhq.swb.de!ora.com!andyo>
Received: from brewhq.swb.de by monad.swb.de with uucp
        (Smail3.1.28.1 #6) id m0pqqlT-00023aB; Wed, 12 Apr 95 00:17 MET DST
Received: from ora.com (ruby.ora.com) by brewhq.swb.de with smtp
        (Smail3.1.28.1 #28.6) id <m0pqoQr-0008qhc>; Tue, 11 Apr 95 21:47 MEST
Received: by ruby.ora.com (8.6.8/8.6.4) id RAA26438; Tue, 11 Apr 95 15:56 -0400
Date: Tue, 11 Apr 1995 15:56:49 -0400
Message-Id: <199404121956.PAA07787@ruby>
```

```
From: andyo@ora.com (Andy Oram)
To: okir@monad.swb.de
Subject: Re: Your RPC section
```

Im Normalfall werden alle notwendigen Einträge im Kopf der Mail von Ihrem Mailprogramm wie *elm*, *pine*, *mush* oder *mailx* generiert. Einige sind optional und können vom Benutzer angefügt werden. *elm* erlaubt Ihnen beispielsweise, einen Teil des Headers zu editieren. Andere Felder werden von der Transport-Software angefügt. Hier ist eine Liste der wichtigsten Header-Einträge und ihrer Bedeutung:

From:
: Dieses Feld enthält die E-Mail-Adresse des Absenders und den »richtigen« Namen. Ein ganzer Zoo an Formaten ist hier erlaubt.

To:
: Dies gibt die E-Mail-Adresse des Empfängers an.

Subject:
: Beschreibt den Inhalt der Nachricht in einigen Worten. Zumindest *sollte* es das tun.

Date:
: Datum und Uhrzeit, zu denen die Nachricht abgeschickt wurde.

Reply-To:
: Dieses Feld ist optional und gibt eine alternative Adresse an, wenn Antworten auf diese Mail an eine andere als die ursprüngliche Adresse des Absenders gehen sollen. Das ist beispielsweise nützlich, wenn Sie verschiedene Accounts besitzen, aber den allergrößten Teil Ihrer Mail auf dem am häufigsten genutzten empfangen wollen.

Organization:
: Die Organisation, der der Rechner gehört, von der die Nachricht verschickt wurde. Wenn es Ihr eigener Rechner ist, lassen Sie das Feld einfach leer, tragen Sie »private« ein oder einen völligen Unsinn. Auch dieses Feld ist optional.

Message-ID:
: Eine eindeutige Kennung, die von der Transport-Software des Absendersystems erzeugt wurde. Sie dient dazu, die Nachricht beispielsweise in Log-Dateien eindeutig identifizieren zu können.

Received:
: Jedes System, das eine Mail bearbeitet (einschließlich der Maschinen von Absender und Empfänger) fügt ein solches Feld in den Header ein, das den Namen des Systems angibt, Datum und Uhrzeit, zu der die Nachricht bearbeitet wurde, von welchem System sie kam und an welches sie weitergereicht wurde. Mit dieser Information können Sie, wenn nötig, die Route zurückverfolgen, die eine Nachricht genommen hat, und im Notfall feststellen, wo das Problem liegt, wenn eine Mail einmal unzustellbar sein sollte.

X-irgendwas:
: Mail-Software sollte sich niemals über Header-Zeilen beschweren, die mit `X-` anfangen. Feldnamen dieser Art können dazu benutzt werden, zusätzliche Dienste zu implementieren, die noch kein Internet-Standard sind und es vielleicht auch niemals werden. Dies wird beispielsweise von der Linux Activists Mailing-Liste ausgenutzt, wo von mehreren sogenannten »Channels« durch das `X-Mn-Key:` Header-Feld ausgewählt werden kann.

Die einzige Ausnahme von dieser Struktur ist die allererste Zeile. Sie beginnt mit dem Schlüsselwort `From`, gefolgt von einem Leerzeichen anstelle eines Doppelpunktes. Um es deutlich vom `From:`-Feld zu unterscheiden, werden wir es im folgenden immer als `From_` bezeichnen. Dieses Feld enthält unter anderem die Route, die die Mail genommen hat, in Bang Path-Notation (wird später erklärt) und die Uhrzeit des Empfangs. Das Feld wird von jeder Maschine, die die Nachricht bearbeitet, regeneriert und wird deshalb oft dem Envelope zugeordnet.

Das `From_`-Feld dient der Rückwärts-Kompatibilität mit älteren Mail-Programmen und wird kaum noch benutzt. Eine Ausnahme bilden Benutzerprogramme wie *elm*, die anhand dieses Felds den Anfang der Nachrichten in der Mailbox-Datei erkennen. Um möglichen Problemen vorzubeugen, hat es sich eingebürgert, Zeilen im Rumpf der Nachricht, die auch mit »From« beginnen, durch ein vorangestelltes »>« zu schützen.

Wie wird Mail transportiert?

Für gewöhnlich werden Sie Ihre elektronischen Briefe mit Hilfe von Programmen wie *mail* oder *mailx* schreiben oder mit etwas komfortableren wie *elm*, *mush* oder *pine*. Diese Programme werden im Englischen als *mail user agents* bezeichnet (*agent* heißt Werkzeug) und MUA abgekürzt. Wenn Sie eine Nachricht abschicken, wird sie von Ihrem Benutzerprogramm an ein anderes Programm weitergereicht, das den Transport zum Zielsystem übernimmt. Dieses Programm heißt *mail transport agent*, kurz MTA. Auf manchen Systemen gibt es je ein Transportprogramm für die lokale Auslieferung und den Transport übers Netz; auf anderen Systemen übernimmt ein Programm beide Aufgaben.

Die Auslieferung einer Mail an einen lokalen Benutzer umfaßt natürlich mehr, als sie nur an die Mailbox-Datei des Benutzers anzuhängen. Normalerweise muß sich der lokale MTA um Aliase (lokale Empfänger, die in Wirklichkeit auf andere Adressen verweisen) und Forwarding (Umleiten aller Mail für einen Benutzer auf ein anderes System oder einen anderen Account) kümmern. Außerdem müssen nicht zustellbare Mails mitsamt einem Fehlerreport an den Absender zurückgesandt werden; im Englischen wird dieser Vorgang als *bouncing* bezeichnet und die generierte Fehlermeldung als *bounce mail*.

Bei der Auslieferung über ein anderes System hängt es ganz von der Art der Netzverbindung ab, welche Transport-Software verwendet wird. Wird die Nachricht über ein TCP/IP-basiertes Netz weitergereicht, wird für gewöhnlich SMTP verwendet. SMTP steht für Simple Mail Transfer Protocol, und wird in RFC 788 und RFC 821 definiert. SMTP baut im allgemeinen eine direkte Verbindung zum Zielrechner auf und überträgt die Nachricht an den SMTP-Dämon der anderen Seite.

In UUCP-Netzen kann Mail dagegen nicht direkt ausgeliefert werden, sondern wird von einer Reihe von Zwischensystemen bis zum Zielrechner weitergereicht. Um eine Nachricht über einen einzelnen UUCP-Link weiterzureichen, ruft der sendende MTA auf dem Nachbarsystem mit Hilfe von *uux* das Programm *rmail* auf und übergibt ihm die Nachricht auf der Standardeingabe.

Da dies für jede Nachricht separat geschieht, kann es auf größeren Mailverteilern zu einer beträchtlichen Arbeitslast kommen und gleichzeitig die UUCP-Spoolverzeichnisse mit Hunderten kleiner Dateien füllen, die überdurchschnittlich viel Plattenplatz belegen.(2) Einige MTAs erlauben Ihnen darum, mehrere Nachrichten, die an dasselbe System weitergereicht werden sollen, in einer Batch-Datei zu sammeln. Diese Batch-Datei enthält die SMTP-Befehle, die der lokale MTA normalerweise ausgeben würde, wenn eine echte SMTP-Verbindung benutzt würde. Diese Technik nennt sich BSMTP, oder *batched* SMTP. Der fertige Batch wird dann auf dem anderen System an das Programm *rsmtp* oder *bsmtp* verfüttert, das die Eingabe verarbeitet, als ob eine normale SMTP-Verbindung vorläge.

E-Mail-Adressen

Eine Adresse für elektronische Mail enthält mindestens den Namen der Maschine, die die Mail des Empfängers bearbeitet, sowie eine Benutzer-Identifikation, die von diesem System erkannt wird. Das ist häufig der Login-Name des Benutzers, kann aber auch etwas völlig anderes sein. Andere Adressierungsmethoden wie X.400 verwenden einen allgemeineren Satz von »Attributen« wie Land und Organisation, aus denen mit Hilfe eines X.500 Directory-Servers das Empfängersystem ermittelt wird.

Die Interpretation eines Rechnernamens, d. h. auf welchem System Ihre Mail am Ende auftauchen wird, und wie dieser Name mit dem Empfängernamen verknüpft wird, hängt stark davon ab, in was für einem Netz Sie sich befinden.

Internet-Systeme halten sich an den Standard RFC 822, der die Schreibweise *user@host.domain* verlangt, wobei *host.domain* der voll qualifizierte Domainname des Zielsystems ist. Das Ding in der Mitte heißt im Deutschen Klammeraffe; der englische Name »at« macht einem eher begreiflich, wieso es als Trennsysmbol gewählt wurde. Da diese Notation nur den Rechnernamen, aber keine Routing-Information angibt, wird diese Form auch gelegentlich als *absolute* Adresse bezeichnet.

In der ursprünglichen UUCP-Umgebung war die vorherrschende Form *path!host!user*, wobei *path* die Reihenfolge der Systeme beschrieb, die die Nachricht zu passieren hatte, bevor sie das Zielsystem **host** erreichte. Die im Pfad angegebenen Systeme werden untereinander ebenfalls durch Ausrufezeichen getrennt. Diese Angabe einer Route nennt sich *bang path*,

da im Amerikanischen ein Ausrufezeichen auch manchmal »bang« genannt wird. Heute haben viele UUCP-basierte Netze RFC 822 angenommen und verstehen auch die RFC-Adressen.

Diese zwei Arten der Adressierung lassen sich allerdings nicht gut mischen. Betrachten Sie die Adresse *hostA!user@hostB*. Es ist nicht ohne weiteres klar, ob das &guilsinglright;@&guilsinglleft;-Zeichen Vorrang vor der Pfadangabe hat, oder umgekehrt: Schicken wir die Nachricht zuerst an **hostB**, von wo sie an *hostA!user* weitergereicht wird, oder muß sie an System **hostA** geschickt werden, das sie an *user@hostB* sendet?

Solche Adressen, die zwei Typen von Adreßoperatoren vermischen, werden *hybride Adressen* genannt. Das oben genannte Beispiel ist das berüchtigste und wird im allgemeinen dadurch gelöst, daß dem @-Zeichen eine höhere Präzedenz eingeräumt wird. Die Mail würde also zuerst an **hostB** geschickt.

Allerdings sieht auch RFC 822 eine Möglichkeit vor, Routen anzugeben: *<@hostA,@hostB:user@hostC>* ist die Adresse von *user* auf **hostC**, wobei **hostC** durch **hostA** und **hostB** (in dieser Reihenfolge) erreicht werden kann. Dies wird oft als *route-addr-Adresse* bezeichnet.

Dann ist da noch der &guilsinglright;%&guilsinglleft;-Operator: Eine Mail an *user%hostB@hostA* wird zuerst an **hostA** geschickt, das das am weitesten stehende (und in diesem Falle einzige) Prozentzeichen durch ein &guilsinglright;@&guilsinglleft;-Zeichen ersetzt. Die Adresse lautet nun *user@hostB*, und das Mail-System wird sie nun fröhlich an **hostB** weiterreichen, das sie an *user* ausliefert. Diese Art der Adressierung wird manchmal als »Ye Olde ARPANET kludge« bezeichnet, und von ihrer Verwendung wird dringend abgeraten. Viele Mail-Transportprogramme erzeugen sie trotzdem.

Andere Netze haben noch ganz andere (und oft nicht weniger barocke) Adressierungsmöglichkeiten. Auf DECnet basierende Netze benutzen beispielsweise zwei Doppelpunkte als Trennzeichen, was Adressen der Form *host::user* ergibt.(3)

Im FidoNet wird jeder Benutzer durch einen Kode wie *2:320/204.9* identifiziert, der aus vier Zahlen besteht, die die Zone bezeichnen (2 für Europa), das Netz (320 für Paris und Banlieue), den Knoten (der lokale Mailverteiler), und den sogenannten Point (der PC des Benutzers). FidoNet-Adressen können auf RFC 822-Adressen abgebildet werden; die obige würde als *Thomas.Quinot@p9.f204.n320.z2.fidonet.org* geschrieben werden. Sagte ich nicht, daß Domainnamen einfach zu merken sind?

Aus den unterschiedlichen Adressierungsarten ergeben sich einige Implikationen für die Mail-Software, die wir im folgenden diskutieren werden. Lassen Sie sich dadurch aber nicht abschrecken; in einer RFC 822-Umgebung werden Sie selten etwas anderes als absolute Adressen wie *user@host.domain* verwenden.

Wie funktioniert Mail-Routing?

Eine zentrale Aufgabe eines MTA ist herauszufinden, wie eine gegebene Nachricht zum Zielsystem transportiert werden kann. Dieser Vorgang heißt Routing. Neben der Suche nach einem Pfad zum Zielsystem gehören dazu die Absicherung gegen Fehler ebenso wie Geschwindigkeits- und Kostenoptimierung.

Es gibt einen großen Unterschied in der Art und Weise, wie ein Internet-System dieses Routing-Problem löst und wie dies ein UUCP-System tut. Im Internet kann eine Nachricht meist direkt an das Zielsystem übertragen werden, da der Transport zum Zielsystem von TCP/IP transparent erledigt wird. Im UUCP-Bereich dagegen muß Ihr Mailer im schlimmsten Falle die genaue Route zum Zielsystem kennen, da die UUCP-Transportprogramme keine Möglichkeit haben, sie selbst zu generieren. Im allgemeinen bedeutet dies, daß die Route entweder vom Benutzer vorgegeben oder vom Transportprogramm selbst erzeugt werden muß.

Mail-Routing im Internet

Im Internet ist es ganz vom Zielsystem abhängig, ob ihr MTA überhaupt ein besonderes Routing vornimmt. Im Normalfall wird eine Nachricht direkt an den Zielrechner übertragen, wobei das tatsächliche Routing ganz der IP-Ebene überlassen wird.

Die meisten Systeme werden es allerdings vorziehen, daß alle eingehenden Mails von einem zentralen Server entgegengenommen und anschließend lokal verteilt werden. Um diesen Server anderen Systemen bekanntzumachen, veröffentlicht dieses System im DNS einen sogenannten MX-Record für seine Domain. MX steht für den englischen Terminus *mail exchanger*. Ein MX-Record besagt, daß der angegebene Server willens und in der Lage ist, Mails an die Domain anzunehmen und weiterzuleiten. MX-Records können deshalb auch dazu verwendet werden, alle Mails für eine Gruppe von Maschinen, die nicht selbst ans Internet angeschlossen sind, auf ein Gateway umzudirigieren. Typische Beispiele sind UUCP- oder firmeninterne Netze, die aus Gründen der Sicherheit vom Internet getrennt sind.

Mit einem MX-Record ist immer eine sogenannte Präferenz verbunden. Die Präferenz ist eine positive Zahl, die angibt, wie bevorzugt Nachrichten über dieses Gateway ausgeliefert werden sollen. Wenn mehrere Gateways für einen Zielrechner existieren, wird der MTA versuchen, die Nachricht an das Gateway mit dem niedrigsten Präferenzwert auszuliefern. Wenn dieser Versuch fehlschlägt, weil beispielsweise die Maschine nicht erreichbar ist, versucht der MTA, die Nachricht an das Gateway mit der nächsthöheren Präferenz auszuliefern. Ist die lokale (ausliefernde) Maschine selbst ein Mail-Exchanger für das Zielsystem, darf sie die Nachricht nur an einen MX mit niedrigerer Präferenz als ihrer eigenen ausliefern. Diese Einschränkung verhindert, daß eine Mail zwischen verschiedenen Gateways Karussell fährt.

Angenommen, eine Firma namens Foobar GmbH möchte, daß alle ankommende Mail von ihrem zentralen Server namens **mailhub** bearbeitet wird. Dann wird sie für jede ihrer Maschinen einen MX-Record wie diesen im DNS eintragen:

```
foobar.com.     IN     MX     5 mailhub.foobar.com.
```

Das macht **mailhub.foobar.com** als Mail-Gateway für die Domain **foobar.com** mit einer Präferenz von 5 bekannt. Ein MTA, der eine Nachricht an *joe@foobar.com* ausliefern will, wird im DNS nach **foobar.com** suchen und obigen MX-Record finden. Wenn er keinen anderen MX-Record mit niedrigerer Präferenz findet, wird er die Nachricht an **mailhub** übertragen, das sienun innerhalb der Firma an joe ausliefert.

Soweit in groben Umrissen, wie der MX-Mechanismus funktioniert. Weitergehende Informationen über das Mail-Routing im Internet finden Sie in RFC 974.

Mail-Routing in der UUCP-Welt

Soweit war ja alles noch recht einfach. Das Mail-Routing in UUCP-Netzen ist allerdings wesentlich komplizierter als im Internet, da die Transport-Software alleine keinerlei Routing durchführen kann. Früher mußten deshalb alle Mail-Adressen als Bang-Pfade angegeben werden. Ein Bang-Pfad gibt eine Liste von Maschinen an, durch die eine Nachricht weitergeleitet werden muß, getrennt durch Ausrufezeichen und gefolgt vom Benutzernamen. Um einen Brief an Janet User auf einer Maschine namens **moria** zu richten, würden Sie eine Adresse wie *eek!swim!moria!janet* angeben müssen. Das würde die Nachricht zuerst an **eek** schicken, von dort an **swim** und schließlich an **moria**, wo sie an **janet** ausgeliefert würde.

Der offensichtliche Nachteil dieser Technik ist, daß Sie dazu genötigt sind, sich viel über die Netztopologie zu merken, wie beispielwsweise schnelle und zuverlässige Verbindungen etc. Schlimmer noch, kurzfristige Veränderungen der Netztopologie wie die Entfernung eines Links oder einer Maschine können dazu führen, daß eine Nachricht ihr Ziel nicht erreicht, einfach weil Ihnen diese Änderung nicht bekannt ist. Zu guter letzt müssen Sie sich im Falle eines Umzugs alle Routen neu zusammensuchen.

Ein wichtiger Grund für die Verwendung solch Absender-abhängiger Adressen (der englische Begriff lautet *source routing*) war, daß Maschinennamen oft nicht eindeutig waren: Nehmen Sie an, es gäbe zwei Systeme namens **moria**; eins in den USA, das andere in Frankreich -- auf welches bezieht sich nun die Adresse *moria!janet*? Die Angabe eines Pfades kann hier eine klare Zuordnung schaffen.

Der erste Schritt in Richtung eindeutiger Systemnamen war die Gründung des UUCP Mapping Project. Das Projekt ist an der Rutgers University (USA) zu Hause und registriert alle offiziellen UUCP-Systeme mitsamt Informationen über die UUCP-Nachbarn jedes Systems und ihrer geographischen Position und stellt sicher, daß kein Name doppelt belegt wird. Die vom Mapping Project gesammelten Informationen werden regeläßig in den *Usenet Maps* veröffentlicht und durch das Usenet verteilt.(4)

Ein typischer Map-Eintrag sieht nach der Entfernung der Kommentare so aus:

```
moria
         bert(DAILY/2),
         swim(WEEKLY)
```

Dieser Eintrag besagt, daß **moria** eine Verbindung zu bert besitzt, die mindestens zweimal täglich aufgebaut wird, sowie eine mit **swim**, mit dem es einmal wöchentlich Mail austauscht. Wir werden uns weiter unten noch etwas ausführlicher mit dem Format der Map-Dateien befassen.

Mit Hilfe der Verbindungsinformationen in den Maps können Sie automatisch den vollen Pfad von Ihrem System zu jedem beliebigen Zielsystem erzeugen. Diese Pfade werden normalerweise in einer Datei namens *paths* oder *pathalias* aufbewahrt. Angenommen, Sie können von Ihrem System aus **bert** über **ernie** erreichen, dann sähe der aus obigem Map-Schnipsel erzeugte pathalias-Eintrag für **moria** so aus:

```
moria     ernie!bert!moria!%s
```

Wenn Sie nun eine Mail an *janet@moria* schicken, wird Ihr MTA die oben gezeigte Route auswählen und die Nachricht mit einer Envelope-Adresse von *bert!moria!janet* an **ernie** schicken.

Es ist allerdings keine besonders gute Idee, eine pathalias-Datei aus den kompletten Usenet Maps zu generieren. Die darin enthaltenen Link-Bewertungen geben die realen Verhältnisse nur sehr ungenau wieder und sind auch nicht immer auf dem neuesten Stand. Aus diesem Grunde erzeugen nur einige sehr große Sites wie **uunet** ihre *paths*-Datei aus den vollen Maps. Die meisten beschränken sich auf Routing-Informationen über Systeme in ihrer Nachbarschaft und leiten Nachrichten an Systeme, die sich nicht in ihren Listen finden, an ein intelligenteres System mit vollständigeren Informationen weiter. Diese Technik heißt im Englischen *smart-host routing*. Sie ist besonders nützlich für Systeme, die nur einen UUCP-Link haben; solche Systeme brauchen sich dann überhaupt nicht mehr um ein intelligentes Routing zu kümmern, sondern überlassen die ganze Arbeit ihrem Smart Host.

UUCP und RFC 822

Die bis heute beste Medizin gegen die Probleme des Mail-Routings in UUCP-Netzen ist die Übernahme von Domainnamen aus dem DNS. Natürlich können Sie über UUCP einen Name-Server nicht nach der optimalen Route zu einem UUCP-System befragen. Trotzdem schließen sich die meisten UUCP-Systeme in Domains zusammen, die ihr Mail-Routing untereinander abstimmen. In den Maps publizieren diese Domains dann ein oder zwei Maschinen als ihre Mail-Gateways, so daß nicht jede einzelne Maschine einen eigenen Map-Eintrag haben muß. Die Gateways bearbeiten alle ein- und ausgehenden Nachrichten, so daß das interne Routing vor der Außenwelt verborgen bleiben kann.

Das funktioniert insbesondere im Zusammenspiel mit dem oben beschriebenen Smart Host-Routing sehr gut. Nur das Gateway selbst muß über globale Routing-Informationen verfügen; kleinere Systeme kommen mit einer kleinen, handgeschriebenen *paths*-Datei aus, die Routen innerhalb der Domain beschreibt. Alle anderen Mails werden an das Gateway weitergereicht. Selbst die Mail-Gateways brauchen jetzt nicht mehr die gesamte Routing-Information für jedes einzelne UUCP-System der Welt bereitzuhalten. Abgesehen von den Informationen über die Topologie ihrer eigenen Domain müssen sie sich nur noch Routen zu ganzen Domains merken. Der folgende pathalias-Eintrag beispielsweise routet alle Mail an Systeme in der Domain **sub.org** zum System **smurf**:

```
.sub.org      swim!smurf!%s
```

Eine Nachricht an *claire@jones.sub.org* wird nun mit einem Envelope vom *smurf!jones!claire* an **swim** weitergereicht werden.

Die hierarchische Organisation des Namensraums im DNS erlaubt es einem Gateway jetzt auch, spezifische Routen mit weniger spezifischen zu mischen. Zum Beispiel kannte seinerzeit das System **unido** in Dortmund alle Routen für UUCP-Domains in Deutschland, aber routete alle Mail an Systeme unterhalb der Domain **nl** (Niederlande) über das System **iafnl.iaf.nl**. Auf diese Art und Weise reduziert das domainorientierte Routing (wie diese Technik genannt wird)

sowohl die Größe der Routing-Tabellen als auch den administrativen Aufwand erheblich.

Der größte Vorteil der Verwendung von Domainnamen in einer UUCP-Umgebung ist aber, daß Adressen nun konform zu RFC 822 sind und damit der Austausch mit dem Internet erheblich einfacher wird. Viele UUCP-Domains haben heute bereits einen direkten Link zu einem Internet-System, das als ihr Relay-System dient. Einerseits ist nämlich der Transport von Mails übers Internet wesentlich schneller, andererseits ist die Routing-Information wesentlich zuverlässiger, da das Gateway anstelle der Maps auf DNS zurückgreifen kann.

Um vom Internet aus erreichbar zu sein, muß eine UUCP-Domain einen MX-Record einrichten lassen, der auf ihr Internet-Gateway zeigt (MX-Records wurden oben im oberen Abschnitt beschrieben). Nehmen Sie beispielsweise an, **moria** gehöre zur Domain **orcnet.org** und **gcc2.groucho.edu** diene als ihr Internet-Gateway. **moria** wird **gcc2** dann als seinen Smart Host benutzen, so daß Nachrichten an Systeme außerhalb der Domain übers Internet ausgeliefert werden. Umgekehrt ist **gcc2** als Mail-Exchanger für **orcnet.org** eingetragen und liefert alle einkommenden Mails an **orcnet**-Systeme über UUCP in die Domain aus.

Das einzige verbleibende Problem ist, daß die UUCP-Transportprogramme mit Domainnamen nichts anfangen können. Die meisten UUCP-Pakete kommen nur mit Namen bis zu acht Zeichen zurecht, einige sogar nur mit weniger; die Benutzung von nicht-alphanumerischen Zeichen wie Punkten steht für die meisten sowieso außer Frage.

Aus diesem Grunde ist eine Abbildung der RFC-Namen auf UUCP-Namen nötig. Unterschiedliche Mail-Pakete gehen dieses Problem aber unterschiedlich an; eine gängige Methode ist, hierfür die pathalias-Datei zu benutzen.

```
moria.orcnet.org     ernie!bert!moria!%s
```

Dies erzeugt aus einer Adresse, die den Domainnamen enthält, einen reinen UUCP-Bang-Pfad. Andere Mail-Programme benutzen hierfür spezielle Dateien; *sendmail* beispielsweise verwendet eine Datei namens *uucpxtable*.

Beim Versenden von Nachrichten aus einem UUCP-Netz ins Internet ist manchmal die umgekehrte Umsetzung notwendig (auch »Domainisieren« genannt). Solange der Absender der Mail den voll qualifizierten Domainnamen in der Zieladresse verwendet, kann dieses Problem vermieden werden, indem die Mail-Software den Domainnamen nicht aus dem Envelope entfernt, sondern unverändert an das Gateway weiterreicht. Allerdings gibt es immer noch UUCP-Sites, die keiner Domain angehören. Sie müssen im allgemeinen durch hybride Formen adressiert werden: Nehmen Sie an, Sie wollten von einem Internet-System aus eine Mail an *big@bird* schreiben und wissen, daß **bird** ein UUCP-Nachbar von **ernie** ist. Dann können Sie beispielsweise die Adresse *big%bird@ernie.sesame.com* verwenden.

Pfade und Landkarten

In UUCP-Netzen stellen pathalias-Dateien die wesentliche Quelle der Routing-Information dar. Dabei kann ein einzelnes System durchaus über mehrere Dateien für unterschiedliche Aufgaben verfügen, beispielsweise je eine für das Routing innerhalb der Domain und über die Domain-Grenze hinweg. Uns geht es im folgenden jedoch nur um das Format dieser Dateien. Ein typischer Eintrag sieht so aus:

```
moria.orcnet.org     ernie!bert!moria!
moria                ernie!bert!moria!%s
```

Hierbei sind der Systemname und die Pfadangabe jeweils durch TABs voneinander getrennt. Dieser Eintrag bewirkt, daß Mails an **moria** über **ernie** und **bert** ausgeliefert werden. Sowohl **moria**s UUCP- als auch Domainname müssen angegeben werden, wenn der MTA keine andere Möglichkeit hat, diese Namen aufeinander abzubilden.

In einer pathalias-Datei können Sie auch Pfade für ganze Domains angeben, indem Sie den Domainnamen mit vorangestelltem Punkt als Zielsystem angeben. Wenn zum Beispiel alle Systeme in **sub.org** über deren Gateway **smurf** erreichbar sind, könnte der entsprechende Eintrag so aussehen:

```
.sub.org             swim!smurf!%s
```

Das Schreiben einer pathalias-Datei ist aber nur zumutbar, wenn Ihr System selbst nicht viel Routing vornehmen muß.

Wenn Sie Mail für eine größere Zahl von Systemen verarbeiten müssen, ist es angebracht, die Pfade mit dem Programm *pathalias* aus einer sogenannten UUCP-Map zu generieren. Maps lassen sich viel leichter auf dem neuesten Stand halten, da Sie ein System einfach dadurch hinzufügen oder entfernen können, indem Sie seinen Map-Eintrag editieren und die Pfade-Informationen neu erzeugen lassen. Obwohl die vom Usenet Mapping Project veröffentlichten Maps kaum noch fürs Routing benutzt werden, ist es für kleinere Netze durchaus noch sinnvoll, ihre Routing-Informationen über ihre eigenen Maps auszutauschen.

Eine Map-Datei besteht im wesentlichen aus einer Liste von Systemen und ihren unmittelbaren Nachbarn. der Systemname beginnt in Spalte eins und wird gefolgt von einer Liste von Links. Die Liste darf sich über mehrere Zeilen erstrecken, wenn die Fortsetzungszeilen mit einem TAB beginnen. Jede Link-Angabe besteht aus dem Namen des Nachbarsystems und den in Klammern angegebenen »Kosten« des Links. Diese Kosten sollen annäherungsweise beschreiben, wie schnell diese Verbindung im Vergleich zu anderen ist, so daß *pathalias* aus mehreren möglichen Pfaden immer den besten auswählen kann. Mit einem Doppelkreuz beginnende Zeilen sind Kommentare und werden ignoriert.

Betrachten wir moria, das zweimal täglich eine Verbindung zu **swim.two.birds** aufbaut und einmal pro Woche mit **bert.sesame.com**. Außerdem ist die Verbindung mit **bert** sehr langsam, da dieses System nur ein 2400bps-Modem verwendet. **moria**s Map-Schnipsel sähe also so aus:

```
moria.orcnet.org
         bert.seame.com(WEEKLY+LOW),
         swim.twobirds.com(DAILY/2)
moria.orcnet.org = moria
```

Die letzte Zeile macht das System auch unter seinem UUCP-Namen bekannt. Beachten Sie, daß es tatsächlich `DAILY/2` heißen muß, da zwei Anrufe täglich die durchschnittliche Laufzeit einer Nachricht über diesen Link tatsächlich halbierten (ganz im Gegensatz zur Telephonrechnung).

Anhand der Informationen aus solchen Maps kann *pathalias* nun einen optimalen Pfad zu jedem in der Datei aufgeführten System finden und in der pathalias-Datei ablegen, die dann direkt von Ihrem MTA verwendet werden kann.

pathalias stellt daneben noch eine ganze Reihe anderer Dienste bereit, wie das »Verstecken« einer Reihe von Systemen hinter einem Gateway (engl.*site hiding*). Diese Fähigkeiten werden heute kaum noch benutzt. Details hierzu entnehmen Sie bitte der Manual-Seite zu *pathalias* ebenso wie eine vollständige Liste der symbolischen Link-Kosten.

Die Kommentare in einer Map-Datei enthalten oft noch zusätzliche Informationen über die beschriebenen Systeme. Diese Angaben müssen in einem sehr rigiden Format gemacht werden, so daß sie später automatisch weiterverarbeitet werden können. Zum Beispiel enthält die *smail*-Distribution Programme, um diese Informationen aufzubereiten und in lesbarer Form darzustellen.

Wenn Sie Ihr UUCP-System in den deutschen Subnet-Maps registrieren, müssen Sie solch einen Map-Schnipsel ausfüllen. Dasselbe gilt, wenn Sie Ihr System beim UUCP Mapping Project registrieren lassen wollen. Als Beispiel sei hier der Eintrag meines Linux-Systems gezeigt:

```
#N      monad, monad.swb.de, monad.swb.sub.org
#S      AT 486DX50; Linux 1.x
#O      private
#C      Olaf Kirch
#E      okir@monad.swb.de
#P      Kattreinstr. 38, D-64295 Darmstadt, FRG
#L      49 52 03 N / 08 38 40 E
#U      brewhq
#W      okir@monad.swb.de (Olaf Kirch); Sun Jul 25 16:59:32 MET DST 1994
#
monad   brewhq(DAILY)
# Domains
monad = monad.swb.de monad = monad.swb.sub.org
```

Der Leerraum nach den ersten zwei Zeichen ist ein TAB. Die Bedeutung der meisten Felder ist recht offensichtlich; eine detaillierte Beschreibung wird regelmäßig in der Newsgruppe de.admin.submaps veröffentlicht. Den größten Spaß macht es, das Feld L auszufüllen: Es gibt Ihre Position in geographischer Länge und Breite an und wird dazu benutzt, die Postscript-Karten zu erzeugen, um die Usenet-Systeme in verschiedenen Ländern sowie weltweit zu zeigen.(5)

Die Konfiguration von elm

elm steht für *electronic mail* und ist eins der eher einleuchtend benannten UNIX-Utilities. Es besitzt eine bildschirmorientierte Benutzerschnittstelle mit einer guten Hilfefunktion. Wir werden uns im folgenden nicht damit befassen, wie Sie *elm* bedienen, sondern uns nur mit seiner Konfiguration beschäftigen.

Theoretisch können Sie *elm* benutzen, ohne sich in irgendeiner Weise um die Konfiguration zu kümmern -- wenn sie Glück haben. Es gibt aber einige Optionen, die Sie unbedingt einstellen müssen, auch wenn sie selten benutzt werden.

Wenn Sie *elm* aufrufen, liest es verschiedene Konfigurations-Variablen aus der Datei *elm.rc* im Verzeichnis */usr/lib*. Anschließend versucht es, *.elm/elmrc* in Ihrem Home-Directory zu lesen. Diese Datei schreiben Sie gewöhnlich nicht selber. *elm* erzeugt sie für Sie, wenn Sie die *elm*-Option »save options« anwählen.

Globale elm-Optionen

Alle Optionen aus der privaten Konfigurations-Datei lassen sich auch im globalen *elm.rc* setzen. Sie gelten dann für alle Benutzer als Voreinstellung und können von Ihnen in ihrem lokalen *elmrc* überschrieben werden.

Daneben hat die globale Datei aber noch eine andere, sehr wichtige Funktion. Sie enthält nämlich Optionen, die den Namen Ihres lokalen Rechners betreffen. In der Virtuellen Brauerei enthält *elm.rc* beispielsweise diese Zeilen:

```
#
# Der lokale Systemname
hostname = vlager
#
# Domainname
hostdomain = .vbrew.com
#
# Voll qualifizierter Domainname (FQDN)
hostfullname = vlager.vbrew.com
```

Diese Optionen legen fest, was sich *elm* unter dem lokalen Systemnamen vorstellt. Obwohl diese Information nur selten benötigt wird, sollten Sie diese Variablen trotzdem anpassen. Es sei darauf hingewiesen, daß diese Angaben nur wirksam werden, wenn sie in der globalen Konfigurations-Datei auftauchen; im privaten *elmrc* werden sie ignoriert.

Nationale Zeichensätze

In letzter Zeit hat es verschiedene Vorschläge gegeben, den RFC 822-Standard dahingehend zu erweitern, daß Mails auch andere Arten von Daten als bloßen Text enthalten können, wie zum Beispiel Postscript-Dateien, Bilder, binäre Daten etc. Die Familie der Standards und RFCs, die diese Aspekte behandelt, wird oft unter dem Begriff MIME zusammengefaßt. MIME heißt *Multipurpose Internet Mail Extensions*, oder Mehrzweckerweiterungen für Internet-Mail. Unter anderem kann mit einer solchen Erweiterung dem Empfänger mitgeteilt werden, ob eine Nachricht Sonderzeichen aus anderen Zeichensätzen als dem US-ASCII Standard enthält, wie zum Beispiel französische Akzente oder deutsche Umlaute. Diese Techniken werden von *elm* in gewissem Umfange unterstützt.

Linux verwendet zur Darstellung von Zeichen intern einen Zeichensatz namens ISO-8859-1. Dieser aparte Name ist die offizielle Bezeichnung eines ISO-Standards; umgangssprachlich ist er auch als Latin1 bekannt. Latin1 ist eine

Obermenge des ASCII-Zeichensatzes und wird häufig zur Darstellung von Sonderzeichen westlicher Sprachen verwendet. Latin1 ist auch deswegen so beliebt, weil es alle Zeichen noch in 8 Bits unterbringt; andere Standards verwenden sogenannte Multi-Byte-Zeichen, bei denen ein Buchstabe oft mehrere Bytes benötigt.

Ein solcher MIME-Standard, RFC 1049, spezifiziert eine Methode, um anzuzeigen, welchen Zeichensatz der Absender benutzt hat. Eine Nachricht, die Symbole aus Latin1 verwendet, sollte im Nachrichtenkopf beispielsweise folgende Zeilen enthalten:

```
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 8bit
```

Das empfangende System sollte diese Zeile auswerten und Sonderzeichen, wenn nötig, in den lokal verwendeten Zeichensatz konvertieren. Für Nachrichten vom Typ `text/plain` hat das Attribut `charset` per Voreinstellung den Wert `us-ascii`.

Die zweite Zeile gibt an, daß die Latin1-Zeichen direkt als 8-Bit-Werte übertragen wurden. Diese Vorgehensweise ist zwar bestechend einfach, hat aber einen großen Nachteil: Die Spezifikation von SMTP sieht nämlich vor, daß Nachrichten nur 7-Bit-Zeichen enthalten dürfen. Viele MTAs sind dieser Vorschrift bis heute auf den Buchstaben treu und löschen beim Bearbeiten jeder Nachricht jeweils das achte Bit, so daß ein **ö** plötzlich zu einem **v** mutiert usw. Dem sollen spezielle Kodierungen wie *quoted-printable* oder *base64* Abhilfe schaffen, die 8-Bit-Zeichen in eine Folge von 7-Bit-Zeichen verschlüsseln. Mit quoted-printable wird aus einem **ö** beispielsweise die Folge **=F6**. Die Verwendung solcher speziellen Kodierungen kann im Encoding-Feld des Nachrichtenkopfes angezeigt werden. Leider werden diese Kodierungen aber nur von sehr wenigen Mail-Programmen unterstützt, so daß der Empfänger einer so behandelten Nachricht oft ratlos vor empfangenen Buchstabensalat steht.

Um Nachrichten mit anderen Zeichensätzen als US-ASCII darstellen zu können, muß *elm* wissen, wie es diese Zeichen ausgeben soll. Wenn es eine Mail mit einem anderen `charset`-Attribut als `us-ascii` erhält (oder einem anderen Typ als `text /plain`), versucht es die Nachricht von dem Programm *metamail* konvertieren und passend ausgeben zu lassen. Mails, die *metamail* benötigen, werden im Übersichts-Menü mit einem `M` in der ersten Spalte angezeigt.

Da Linux von Hause aus ISO-8859-1 versteht, ist es nicht nötig, Nachrichten, die Latin1 verwenden, gesondert zu behandeln. Wenn Sie folgende Option in Ihrem globalen *elm.rc* setzen, wird *elm* solche Mails direkt darstellen und auf *metamail* verzichten:

```
displaycharset = iso-8859-1
```

Damit ist es allerdings noch nicht getan. Wenn *elm* eine Nachricht mit seinem eingebauten Pager darstellt, ruft es für jedes Zeichen eine Bibliotheks-Funktion auf, um festzustellen, ob es druckbar ist oder nicht. Im Normalfall erkennt diese Funktion nur ASCII-Zeichen als druckbar an, so daß *elm* alle Umlaute nun als »**^ ?**« ausgibt. Dieses Problem räumen Sie dadurch aus der Welt, daß Sie die Umgebungsvariable `LC_CTYPE` auf den Wert `ISO-8859-1` setzen. Das teilt der Bibliothek mit, daß sie Latin1-Buchstaben als druckbar akzeptieren soll. Die Unterstützung für diese (und verwandte Funktionen) ist seit Version 4.5.8 der C-Bibliothek vorhanden.

Wenn Sie in Ihren eigenen Mails Umlaute oder ähnliche Sonderzeichen verwenden, sollten Sie sicherstellen, daß in Ihrem *elm.rc* die folgenden zwei Variablen gesetzt sind:

```
charset = iso-8859-1
textencoding = 8bit
```

Das veranlaßt *elm*, im `Content-Type`-Feld des Nachrichtenkopfes das `charset`-Attribut auf `iso-8859-1` zu setzen und alle Zeichen als 8-Bit-Werte zu übertragen. Im Normalfall löscht *elm* vor dem Versenden einer Mail nämlich immer das oberste Bit, was alle Sonderzeichen zerstören würde.

Fußnoten

(1)

Es ist auch üblich, eine Art Unterschrift an alle Mails anzuhängen, die Informationen über den Absender enthält, zusammen mit einem Witz oder einem Motto. Diese signature oder .sig wird vom Rest der Botschaft durch eine einzelne Zeile abgesetzt, die die Zeichen » « enthält.

(2)

Das kommt daher, daß Plattenplatz in größeren Einheiten wie 1024 Bytes alloziert wird. Dadurch verschwendet jede Mail im Durchschnitt 512 Byte Plattenspeicher.

(3)

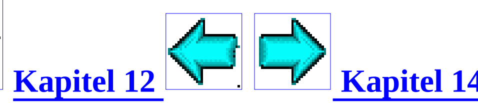
Wenn Sie aus einem Netz mit RFC-Adressierung eine DECnet-Adresse erreichen wollen, können Sie "host::user"@relay verwenden, wobei relay ein Internet-DECnet-Gateway sein muß.

(4)

Die Maps für vom Projekt registrierte Systeme werden in der Newsgruppe comp.mail.maps gepostet. Andere Organisationen veröffentlichen unter Umständen ihre eigenen Maps. Die Liste aller deutschen UUCP-Systeme im Subnetz und im Individual Network wird über die Newsgruppe de.admin.submaps verteilt.

(5)

Sie werden regelmäßig in news.lists.ps-maps veröffentlicht. Seien Sie vorsichtig--sie sind sehr GROSS.

Inhaltsverzeichnis  Kapitel 12  Kapitel 14

Kapitel 14
smail zum Laufen bringen

Dieses Kapitel enthält eine kurze Einführung zur Einrichtung von *smail* und eine Übersicht der gebotenen Funktionalität. Obwohl *smail* von seinem Verhalten her größtenteils kompatibel mit *sendmail* ist, sind die Konfigurations-Dateien doch völlig unterschiedlich.

Die Haupt-Konfigurationsdatei ist */usr/lib/smail/config*. Diese Datei enthält die für Ihre Site spezifischen Werte. Wenn Sie nur eine UUCP-Site betreiben, die keine weiteren Server mit Daten versorgt, bleibt für Sie nicht viel mehr zu tun. Andere Dateien, mit denen die Routing- und Transport-Optionen konfiguriert werden, sind ebenfalls vorhanden und werden kurz besprochen.

Standardmäßig verarbeitet *smail* die gesamte eingehende Post sofort und liefert sie auch direkt aus. Wenn Sie relativ viel Datenverkehr haben, können Sie *smail* auch anweisen, alle Nachrichten in der sog. *Queue* zu speichern und sie nur in regelmäßigen Intervallen zu bearbeiten.

Wenn Sie Mail in einem TCP/IP-Netzwerk verarbeiten, wird *smail* häufig im Dämon-Modus ausgeführt: Während der Bootphase des Systems wird er aus *rc.inet2* heraus gestartet, begibt sich selbst in den Hintergrund und wartet auf eingehende TCP-Verbindungen auf dem SMTP-Port (üblicherweise Port 25). Diese Lösung ist besonders vorteilhaft, wenn Sie mit einem großen Datenvolumen rechnen, weil *smail* dann nicht für jede eingehende Verbindung separat gestartet werden muß. Alternativ könnte *inetd* den SMTP-Port überwachen und jedesmal *smail* starten, wenn eine Anforderung über diesen Port kommt.

smail besitzt verschiedene Optionen, mit denen Sie sein Verhalten kontrollieren können. Sie an dieser Stelle zu beschreiben, würde wenig Sinn machen. Glücklicherweise unterstützt *smail* eine Reihe von Standard-Betriebsmodi, die automatisch aktiviert werden, wenn Sie das Programm unter einem besonderen Programmnamen wie *rmail* oder *smtpd* aufrufen. Normalerweise handelt es sich bei diesen Aliases immer um symbolische Links auf *smail* selbst. Sie werden die meisten bei der Besprechung der verschiedenen *smail*-Features kennenlernen.

Es gibt zwei Links auf *smail*, die Sie unter allen Umständen haben sollten: */usr/bin /rmail* und */usr/sbin/sendmail*.(1) Wenn Sie Nachrichten mit einem Benutzer-Frontend wie *elm* erstellen und

verschicken, wird die Nachricht für die Auslieferung über eine Pipe an *rmail* durchgereicht, wobei die Empfängerliste in der Kommandozeile mit übergeben wird. Dasselbe geschieht mit Mail, die über UUCP eingeht. Dabei starten einige Versionen von elm */usr/sbin/sendmail* anstelle von *rmail*, d. h. beide müssen vorhanden sein. Befindet sich *smail* beispielsweise in */usr/local/bin*, geben Sie den folgenden Befehl ein:

```
# ln -s /usr/local/bin/smail /usr/bin/rmail
# ln -s /usr/local/bin/smail /usr/sbin/sendmail
```

Wenn Sie tiefer in die Details der Konfiguration von *smail* einsteigen wollen, wenden Sie sich an die Manpages *smail(1)* und *smail(5)*. Sind diese in Ihrer Linux-Distribution nicht enthalten, finden Sie sie über die *smail*-Quellen.

UUCP-Setup

Soll *smail* in einer reinen UUCP-Umgebung benutzt werden, ist die grundlegende Installation einfach. Stellen Sie zuerst einmal sicher, daß die beiden oben besprochenen symbolischen Links auf *rmail* und *sendmail* existieren. Wenn Sie SMTP-Batches von anderen Sites erwarten, müssen Sie auch *rsmtp* als Link auf *smail* definieren.

Vince Skahans *smail*-Distribution enthält eine Beispiel-Konfigurationsdatei. Sie heißt *config.sample* und ist in */usr/lib/smail* zu finden. Kopieren Sie sie nach *config* und überarbeiten Sie sie so, daß die für Ihre Site entsprechenden Werte enthalten sind.

Gehen wir davon aus, daß Ihr System **swim.twobirds.com** heißt und in den UUCP-Maps als **swim** registriert ist. Der »Smart Host« ist ulysses. Ihre *config*-Datei sollte dann folgendermaßen aussehen:

```
#
# unser Domainname
visible_domain=two.birds:uucp
#
# unser Name bei ausgehenden Mails
visible_name=swim.twobirds.com
#
# ist gleichzeitig auch der UUCP-Name
uucp_name=swim.twobirds.com
#
# unser "Smart Host"
smart_host=ulysses
```

Die erste Zeile teilt *smail* mit, zu welcher Domain Ihre Site gehört. Geben Sie die entsprechenden Namen, durch Doppelpunkte voneinander getrennt, hier an. Ist Ihr Sitename in den UUCP-Maps registriert, sollten Sie auch `uucp` hinzufügen. Erhält *smail* eine E-Mail, ermittelt es mit dem Systemaufruf *hostname* Ihren Hostnamen und vergleicht die Adresse des Empfängers mit diesem Hostnamen, wobei nacheinander alle Namen der Liste angehangen werden. Stimmt die Adresse mit einem dieser Namen oder mit dem unqualifizierten Hostnamen überein, wird davon ausgegangen, daß sich der Empfänger auf diesem

Rechner befindet. *smail* versucht daraufhin, die Nachricht an einen Benutzer oder Alias auf dem lokalen Host weiterzuleiten. Anderenfalls wird davon ausgegangen, daß der Empfänger auf einem anderen Host zu finden ist, und die Nachricht wird an einen anderen Host weitergeleitet.

`visible_name` sollte einen einzelnen, voll qualifizierten Domainnamen für Ihre Site enthalten, der bei ausgehenden Mails verwendet werden soll. Dieser Name wird bei der Generierung der Absender-Adresse aller ausgehenden Mails verwendet. Sie müssen einen Namen verwenden, den *smail* als einen für den lokalen Host gültigen akzeptiert (z. B. den Hostnamen gemeinsam mit einer der unter `visible_domain` aufgelisteten Domains). Anderenfalls können Antworten Ihre Site niemals erreichen.

Die letzte Zeile setzt den Pfad, der zum »Smart Host«-Routing verwendet wird (wird im vorangehenden Kapitel beschrieben). Bei diesem Beispiel-Setup leitet *smail* alle Nachrichten an entfernte Adressen an den sog. »Smart Host« weiter. Der bei `smart_path` angegebene Pfad wird dabei als Route zu diesem Host verwendet. Weil Nachrichten mittels UUCP ausgeliefert werden, muß ein System angegeben werden, das Ihrer UUCP-Software bekannt ist. Wie Sie UUCP eine Site bekanntgeben, ist in Kapitel 12, *Verwalten von TaylorUUCP* beschrieben.

Eine Option wurde bisher noch nicht erklärt, nämlich `uucp_name`. Der Grund für die Verwendung dieser Option liegt darin, daß *smail* standardmäßig den von *hostname* zurückgelieferten Wert für UUCP-spezifische Informationen verwendet, wie beispielsweise den in der `From_`-Headerzeile vorhandenen Rückgabepfad. Ist Ihr Hostname *nicht* beim UUCP Mapping Project registriert, sollte *smail* statt dessen den voll qualifizierten Domainnamen verwenden.(2) Dies können Sie durch Verwendung der Option `uucp_name` in der *config*-Datei realisieren.

Eine andere Datei in */usr/lib/smail* ist *paths.sample*. Sie stellt ein Beispiel dafür dar, wie eine *paths*-Datei aussehen könnte. Allerdings benötigen Sie diese Datei nur, wenn Sie Mail-Links zu mehr als einer Site besitzen. Wenn dies so ist, müssen Sie allerdings Ihre eigene Datei schreiben oder eine aus den Usenet Maps generieren. Die *paths*-Datei wird später in diesem Kapitel beschrieben.

LAN-Setup

Wenn Sie eine Site mit zwei oder mehreren Hosts betreiben, die über ein LAN miteinander verbunden sind, müssen Sie einen Host bestimmen, der die UUCP-Verbindung mit der Außenwelt hält. Zwischen den einzelnen Hosts in Ihrem LAN werden Sie Mail üblicherweise mit SMTP über TCP/IP austauschen. Wenden wir uns noch einmal unserer virtuellen Brauerei zu, und nehmen wir einmal an, daß **vstout** als UUCP-Gateway verwendet wird.

In einer Netzwerkumgebung ist es am besten, alle Benutzer-Mailboxen auf einem einzelnen System zu halten, das über NFS an alle anderen Hosts gemountet wird. Das erlaubt es den Benutzern, sich an einer beliebigen Maschine einzuloggen, ohne die Post immer mitnehmen zu müssen (oder, schlimmer noch, jeden Morgen drei oder vier Maschinen nach neu eingegangener Post überprüfen zu müssen). Aus diesem Grund sollte die Adresse des Absenders auch unabhängig von der Maschine sein, auf der sie geschrieben wurde. Es ist gängige Praxis, den reinen Domainnamen in der Adresse zu verwenden und nicht den Hostnamen. Janet beispielsweise würde *janet@vbrew.com* spezifizieren und nicht *janet@vale.vbrew.com*. Nachfolgend erklären wir, wie Sie den Server dazu bringen, den Domainnamen als gültigen Namen für Ihre Site zu verwenden.

Eine andere Möglichkeit, alle Mailboxen auf einem zentralen Host zu halten, bietet die Verwendung von POP oder IMAP. POP steht für *Post Office Protocol* und erlaubt es Benutzern, über eine einfache TCP/IP-Verbindung auf deren Mailboxen zuzugreifen. IMAP, das *Interactive Mail Access Protocol*, ist ähnlich wie POP, aber etwas allgemeiner gehalten. Clients und Server für IMAP und POP wurden auf Linux portiert und sind auf sunsite.unc.edu unter */pub/Linux/system/Network* zu finden.

Schreiben der Konfigurations-Dateien

Die Konfiguration für die Brauerei stellt sich dar wie folgt. Alle Hosts außer dem Mail-Server (**vstout**) selbst routen alle ausgehende Post an diesen Server mit Hilfe des »Smart Host«-Routing. **vstout** selbst sendet die gesamte ausgehende Post an den eigentlichen »Smart Host«, der die gesamte Mail der Brauerei routet. Der Host hat den Namen moria.

Die *config*-Datei für alle Hosts außer **vstout** sieht also so aus:

```
#
# unsere Domain
visible_domain=vbrew.com
#
# wie wir uns selbst nennen
visible_name=vbrew.com
#
# Smart Host-Routing: über SMTP an vstout
smart_path=vstout
smart_transport=smtp
```

Das ist dem sehr ähnlich, was wir für die reine UUCP-Site verwendet haben. Der Hauptunterschied besteht darin, daß für den Transport der zu sendenden E-Mails an den Smart Host nun natürlich SMTP verwendet wird. Durch das Attribut `visible_domain` verwendet *smail* den Domainnamen anstelle des lokalen Hostnamens für die gesamte ausgehende Post.

Auf dem UUCP-Mail-Gateway **vstout** sieht die *config*-Datei ein wenig anders aus:

```
#
# unsere Hostnamen
hostnames=vbrew.com:vstout.vbrew.com:vstout
#
# wie wir uns selbst nennen
visible_name=vbrew.com
#
# in der UUCP-Welt sind wir als vbrew.com bekannt
uucp_name=vbrew.com
#
# Smart Transport: über UUCP an moria
smart_path=moria
smart_transport=uux
#
```

```
# wir sind für unsere Domain verantwortlich
auth_domains=vbrew.com
```

Diese *config*-Datei verwendet ein anderes Schema, um *smail* mitzuteilen, wie der lokale Host genannt wird. Statt eine Liste mit Domains zur Verfügung zu stellen und den Hostnamen über einen Systemaufruf ermitteln zu lassen, wird eine explizite Liste angegeben. Die obige Liste enthält den voll qualifizierten sowie den unqualifizierten Hostnamen und den Domainnamen selbst. Auf diese Weise erkennt *smail* *janet@vbrew.com* als lokale Adresse und liefert die Nachricht an *janet* aus.

Die Variable `auth_domains` bestimmt die Domains, für die **vstout** »verantwortlich« zeichnet. Erhält *smail* eine an **host.vbrew.com** adressierte Mail, wobei **host** aber keine existierende lokale Maschine benennt, lehnt es die Nachricht ab und schickt sie zurück an den Absender. Ist dieser Eintrag nicht vorhanden, wird jede solche Nachricht an den Smart Host zurückgeschickt, der sie wieder an **vstout** schickt und so weiter. (Glücklicherweise gibt es ein Feature namens *Maximum Hop Count*, mit dem die Nachricht irgendwann doch ausrangiert wird.) D

smail betreiben

Zuerst müssen Sie entscheiden, ob *smail* als separater Dämon laufen soll, oder ob *inetd* den SMTP-Port verwalten soll und *smail* nur ausführt, wenn eine SMTP-Verbindung von einem Client angefordert wird. Üblicherweise ziehen Sie den Dämon-Betrieb auf dem Mail-Server vor, weil das die Maschine wesentlich weniger belastet als der wiederholte Start von *smail* bei jeder neuen Verbindung. Weil der Mail-Server auch die meiste eingehende Post direkt an die Benutzer verteilt, wird der *inetd*-Betrieb für die meisten anderen Hosts gewählt.

Welchen Betriebsmodus Sie für jeden einzelnen Host auch wählen, Sie müssen die folgende Zeile immer in Ihrer */etc/services*-Datei stehen haben:

```
smtp            25/tcp          # Simple Mail Transfer Protocol
```

Das definiert die TCP-Port-Nummer, die *smail* zur SMTP-Kommunikation verwenden soll. Port 25 ist der im »Assigned Numbers RFC« definierte Standard.

Im Dämon-Modus geht *smail* selbständig in den Hintergrund und wartet darauf, daß eine Anforderung auf dem SMTP-Port eingeht. Taucht eine solche Anforderung auf, wird eine SMTP-Kommunikation mit dem anderen Prozeß aufgebaut und durchgeführt. Der *smail*-Dämon wird normalerweise mit dem folgenden Befehl aus dem *rc.inet2*-Script heraus gestartet:

```
/usr/local/bin/smail -bd -q15m
```

Die Option *-bd* aktiviert den Dämon-Modus. Mit *-q15m* werden alle Nachrichten, die in der Queue aufgelaufen sind, alle 15 Minuten bearbeitet.

Wollen Sie statt dessen mit *inetd* arbeiten, sollte Ihre */etc/inetd.conf* die folgende Zeile enthalten:

```
smtp    stream  tcp nowait  root  /usr/sbin/smtpd smtpd
```

smtpd sollte dabei ein symbolischer Link auf die *smail*-Binary sein. Denken Sie daran, daß *inetd* die *inetd.conf* erneut lesen muß. Senden Sie ihm nach Durchführung der Änderungen einfach ein `HUP`-Signal.

Dämon- und *inetd*-Modus schließen sich gegenseitig aus. Wird *smail* im Dämon-Modus betrieben, müssen Sie sicherstellen, daß in *inetd.conf* alle Zeilen für den `smtp`-Service entfernt wurden. Wird *smail* dagegen über *inetd* verwaltet, muß sichergestellt sein, daß in *rc.inet2* der *smail*-Dämon nicht gestartet wird.

Wenn Ihre Mail nicht durchkommt

Eine Reihe von Features sind vorhanden, um die Fehlersuche bei Installations-Problemen zu erleichtern. Überprüfen Sie zuerst die *smail*-Logdateien. Diese sind in */var/spool/smail/log* zu finden und heißen *logfile* und *paniclog*. Die erste führt alle Transaktionen auf, während die zweite nur Fehlermeldungen enthält, die mit Konfigurations-Fehlern und ähnlichem zu tun haben.

Ein typischer Eintrag in *logfile* sieht wie folgt aus:

```
04/24/94 07:12:04: [m0puwU8-00023UB] received
|             from: root
|          program: sendmail
|             size: 1468 bytes
04/24/94 07:12:04: [m0puwU8-00023UB] delivered
|              via: vstout.vbrew.com
|               to: root@vstout.vbrew.com
|          orig-to: root@vstout.vbrew.com
|           router: smart_host
|        transport: smtp
```

Dies zeigt, daß eine Nachricht von **root** an *root@vstout.vbrew.com* ordnungsgemäß an den Host **vstout** über SMTP ausgeliefert wurde.

Nachrichten, die *smail* nicht ausliefern konnte, erzeugen einen ähnlichen Eintrag in der Log-Datei, statt *delivered* ist aber eine Fehlermeldung zu sehen:

```
04/24/94 07:12:04: [m0puwU8-00023UB] received
|             from: root
|          program: sendmail
|             size: 1468 bytes
04/24/94 07:12:04: [m0puwU8-00023UB] root@vstout.vbrew.com ... deferred
 (ERR_148) transport smtp: connect: Connection refused
```

Der obige Fehler ist typisch für eine Situation, in der *smail* richtig erkannt hat, daß die Nachricht an **vstout** ausgeliefert werden soll, aber nicht in der Lage ist, eine Verbindung zum SMTP-Service auf **vstout** herzustellen. Wenn dies geschieht, haben Sie entweder ein Konfigurations-Problem, oder Ihre *smail*-Binaries besitzen keine TCP-Unterstützung.

Dieses Problem ist nicht so selten, wie man vielleicht denken könnte. Es sind durchaus einige vorkompilierte *smail*-Binaries ohne TCP/IP-Unterstützung in Umlauf, sogar in einigen Linux-Distributionen. Wenn dies bei Ihnen der Fall ist, müssen Sie *smail* selbst kompilieren. Sie können prüfen, ob Ihr *smail* TCP-Netze unterstützt, indem Sie mit Telnet auf den SMTP-Port Ihrer Maschine zugreifen. Eine erfolgreiche Verbindung zum SMTP-Server sieht so aus:

```
$ telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 monad.swb.de Smail3.1.28.1 #6 ready at Sun, 23 Jan 94 19:26 MET
QUIT
221 monad.swb.de closing connection
```

Wenn dieser Test nicht das SMTP-Banner erzeugt (die mit dem Kode 220 beginnende Zeile), sollten Sie auf jeden Fall ganz sicher sein, daß Ihre Konfiguration *wirklich* korrekt ist, bevor Sie sich an die Neukompilierung von *smail* wagen, die nachfolgend beschrieben wird.

Tritt ein Problem mit *smail* auf, das Sie auch durch die von *smail* generierten Fehlermeldungen nicht lokalisieren können, besteht die Möglichkeit, sich Debugging-Nachrichten ausgeben zu lassen. Sie aktivieren dies durch Verwendung der Option *-d*. Dieser kann optional eine Zahl folgen, die bestimmt, wie umfangreich die Meldungen sind (zwischen der Option selbst und der Zahl darf kein Leerzeichen stehen). *smail* gibt dann einen Bericht über seine Operationen auf dem Bildschirm aus, der Ihnen weitere Hinweise auf den Fehler geben kann.

Wenn nichts mehr hilft, können Sie *smail* mit der Option *-bR* im Rogue-Modus starten. Die Manpage sagt zu dieser Option: »Enter the hostile domain of giant mail messages, and RFC standard scrolls. Attempt to make it down to protocol level 26 and back.« Wenn diese Option Ihre Probleme auch nicht lösen wird, so mag sie Ihnen doch ein wenig Trost und Beistand spenden. ;-)

smail kompilieren

Wenn Sie sich sicher sind, daß *smail* keine TCP-Netzwerk-Unterstützung besitzt, müssen Sie sich den Quellkode besorgen. Dieser ist möglicherweise bereits in Ihrer Distribution enthalten. Anderenfalls können Sie sich den Quellkode über FTP aus dem Netz besorgen.(3)

Bei der Kompilierung von *smail* sollten Sie mit dem Satz von Konfigurations-Dateien aus Vince Skahans *newspak*-Distribution beginnen. Um den TCP-Netzwerktreiber zu aktivieren, müssen Sie das `DRIVER_CONFIGURATION`-Makro in der Datei *conf /EDITME* entweder auf `bsd-network` oder auf `arpa-network` setzen. Ersteres ist für LAN-Installationen geeignet, für das Internet wird aber `arpa-network` benötigt. Der Unterschied zwischen diesen beiden besteht darin, daß `arpa-network` einen speziellen Treiber für den BIND-Service besitzt, der in der Lage ist, MX-Records zu erkennen, was

`bsd-network` nicht kann.

Mail-Auslieferungsmodi

Wie bereits oben erwähnt, ist *smail* in der Lage, Nachrichten direkt auszuliefern oder in einer Queue zur späteren Bearbeitung zwischenzuspeichern. Verwenden Sie die Message-Queue, speichert *smail* alle Nachrichten im *messages*-Verzeichnis unter */var/spool/smail*. Die Nachrichten werden nicht bearbeitet, bis dazu die explizite Aufforderung erfolgt (dies wird auch als »Starten der Queue« bezeichnet).

Sie können einen von drei vorhandenen Auslieferungs-Modi wählen, indem Sie das Attribut `delivery_mode` in der Datei *config* einstellen. Die möglichen Werte sind `foreground`, `background` oder `queued`. Damit können Sie die Auslieferung im Vordergrund (direkte Bearbeitung aller eingehenden Nachrichten), im Hintergrund (Die Nachricht wird durch einen Kind-Prozeß des empfangenden Prozesses ausgeliefert; dabei wird der Parent-Prozeß sofort nach dem Start wieder beendet) oder über die Queue wählen. Alle eingehenden Mails werden über die Queue verarbeitet, wenn die Boolesche Variable `queue_only` in der *config*-Datei gesetzt ist. Dabei wird der eingestellte Modus ignoriert.

Wenn Sie Queues benutzen, müssen Sie auch sicherstellen, daß sie regelmäßig, sagen wir alle 10 bis 15 Minuten, überprüft werden. Wird *smail* im Dämon-Modus betrieben, müssen Sie die Option *-q10m* in der Kommandozeile angeben, damit die Queue alle 10 Minuten abgearbeitet wird. Alternativ können Sie *runq* aus *cron* heraus in diesen Intervallen starten. *runq* sollte ein Link auf *smail* sein.

Sie können sich die aktuelle Mail-Queue durch Ausführen von *smail* mit der Option *-bp* ansehen. Sie können auch einen Link auf *smail* namens *mailq* einrichten und *mailq* aufrufen:

```
$ mailq -v
m0pvB1r-00023UB  From: root  (in /var/spool/smail/input)
                Date: Sun, 24 Apr 94 07:12 MET DST
                Args: -oem -oMP sendmail root@vstout.vbrew.com
Log of transactions:
 Xdefer: <root@vstout.vbrew.com> reason: (ERR_148) transport smtp:
 connect: Connection refused
```

Dies zeigt eine einzelne, in der Message-Queue vorliegende Nachricht. Der Transaktionslog (der nur ausgegeben wird, wenn Sie *mailq* mit der Option *-v* aufrufen) gibt unter Umständen auch an, warum die Nachricht noch nicht ausgeliefert wurde. Wurde bislang noch kein Versuch unternommen, die Nachricht auszuliefern, wird kein Transaktionslog ausgegeben.

Selbst wenn Sie nicht mit der Queue arbeiten, wird *smail* gelegentlich Nachrichten in die Queue schieben, nämlich dann, wenn die unmittelbare Auslieferung aus irgendeinem Grund fehlgeschlagen ist. Bei SMTP-Verbindungen könnte der Grund darin liegen, daß der Host momentan nicht erreichbar ist, aber auch, daß dessen Dateisystem gerade voll ist. Daher sollten Sie die Queue ungefähr stündlich ausführen lassen (mit *runq*), weil sonst solche zwischengespeicherten Nachrichten für immer in der Queue bleiben.

Verschiedene config-Optionen

Hier noch einige nützliche Optionen, die Sie in Ihrer config-Datei eintragen können:

`error_copy_postmaster`

Ist diese Boolesche Variable gesetzt, wird bei jedem Fehler eine Meldung an den Postmaster erzeugt. Normalerweise tut smail dies nur, wenn es über einen Konfigurationsfehler stolpert. Sie schalten diese Variable ein, indem Sie sie in Ihre *config*-Datei einfügen und ihr ein Plus (`+`) voranstellen.

`max_hop_count`

Wenn der Sprungzähler (Hop Count, d. h. die Anzahl der bereits passierten Hosts) für eine Nachricht größer oder gleich dieser Zahl ist, führt der Versuch, diese Nachricht erneut auszuliefern, zu einer Fehlermeldung, die an den Absender geschickt wird. Auf diese Weise wird verhindert, daß Nachrichten bei der Auslieferung in einer Endlosschleife landen. Der Hop Count wird üblicherweise aus der Zahl der `Received:`-Felder im Mail-Header ermittelt. Mit der Option *-h* kann er in der Kommandozeile aber auch von Hand eingegeben werden.

Diese Variable ist auf den Wert 20 voreingestellt.

`postmaster`

Die Adresse des Postmasters. Kann die Adresse *Postmaster* nicht in eine gültige lokale Adresse aufgelöst werden, wird diese als letzte Rettung benutzt. Standardmäßig zeigt dieser Wert auf *root*.

Routing und Auslieferung von Nachrichten

smail teilt die Auslieferung von Mails in drei verschiedene Untersysteme auf: das Router-, Director- und das Transport-Modul.

Das Router-Modul löst alle entfernten Adressen auf und ermittelt dabei, an welchen Host die Nachricht als nächstes weitergeleitet werden soll, und wie der Transport zu erfolgen hat. Abhängig von der Natur des Links, können unterschiedliche Transportmechanismen wie UUCP oder SMTP verwendet werden.

Lokale Adressen werden an das Director-Modul übergeben, das alle Weiterleitungs- bzw. Aliasing-Aufgaben übernimmt. So könnte die Adresse etwa ein Alias oder eine Mailing-Liste sein, oder ein Benutzer könnte seine Mails auf eine andere Adresse umleiten. Ist die ermittelte Adresse nicht lokal, wird sie an das Router-Modul zum weiteren Routing übergeben. Anderenfalls wird sie zur lokalen Auslieferung an das Transport-Modul übergeben. Der bei weitem am häufigsten auftretende Fall ist, daß die Nachricht in eine Mailbox ausgeliefert wird. Nachrichten können aber über eine Pipe auch an einen Befehl übergeben oder an eine bestimmte Datei angehängt werden.

Das Transport-Modul ist dafür verantwortlich, welche Methode der Auslieferung verwendet wird. Es versucht, die Nachricht auszuliefern und erzeugt im Fehlerfall entweder eine Bounce-Nachricht (d. h. die Mail wird an den Absender zurückgeschickt), oder versucht es zu einem späteren Zeitpunkt erneut.

Mit *smail* haben Sie bei der Konfiguration dieser Aufgaben viele Freiheiten. Für jede steht eine ganze Reihe von Treibern bereit, unter denen Sie sich den benötigten aussuchen können. Sie beschreiben sie *smail* gegenüber durch eine Reihe von Dateien, nämlich *routers*, *directors* und *transports*, die unter

/usr/lib/smail zu finden sind. Sind diese Dateien nicht vorhanden, werden vernünftige Standardwerte als gegeben angenommen, die für viele Sites passend sind, die entweder SMTP oder UUCP für den Transport benutzen. Wollen Sie die Routing- oder Transport-Methode von *smail* modifizieren, sollten Sie sich die Beispieldateien aus der *smail*-Source-Distribution(4) besorgen, die Beispieldateien nach */usr/lib/smail* kopieren und sie entsprechend Ihren Bedürfnissen anpassen. Schlagen Sie in Anhang B nach.

Nachrichten routen

Trifft eine Nachricht ein, prüft *smail* zuerst, ob das Ziel der lokale Host oder eine andere Site ist. Ist die Host-Zieladresse einer der lokalen, in *config* konfigurierten Hostnamen, wird die Nachricht an das Director-Modul übergeben. Anderenfalls leitet *smail* die Zieladresse an eine Reihe von Router-Treibern weiter, um herauszufinden, an welchen Host eine Nachricht weitergeleitet werden soll. Diese Treiber können in der Datei *routers* beschrieben werden; falls diese Datei nicht existiert, wird ein Satz von Standard-Routern verwendet.

Der Zielhost wird an alle Router übergeben, und der Router, der die spezifischste Route findet, wird gewählt. Nehmen wir beispielsweise eine an *joe@foo.bar.com* adressierte Nachricht. Ein Router könnte eine Standard-Route für alle Hosts in der **bar.com**-Domain kennen, während ein anderer die Information zu **foo.bar.com** selbst besitzt. Weil letzterer spezifischere Informationen besitzt, würde er gegenüber dem ersten den Vorrang erhalten. Liefern zwei Router ein gleich gutes Resultat zurück, wird derjenige gewählt, der in der Datei *routers* als erster erscheint.

Der Router spezifiziert nun die zu verwendende Transportmethode, beispielsweise UUCP, und generiert eine neue Zieladresse. Die neue Adresse wird zusammen mit dem Host, an den die Nachricht weitergeleitet werden soll, an das Transport-Modul übergeben. Im obigen Beispiel findet *smail* heraus, daß **foo.bar.com** mit UUCP über den Pfad *ernie!bert* erreicht werden kann. Es generiert dann *bert!foo.bar.com!user* als neues Ziel und läßt den UUCP-Transport diesen Ausdruck als die an ernie zu übergebende Adresse verwenden.

Wenn Sie das Standard-Setup benutzen, stehen die folgenden Router zur Verfügung:

- Kann die Host-Zieladresse mit den Bibliotheksfunktionen *gethostbyname* oder *gethostbyaddr* aufgelöst werden, wird die Nachricht über SMTP ausgeliefert. Die einzige Ausnahme tritt auf, wenn die Adresse einen lokalen Host bezeichnet, wobei sie an das Director-Modul übergeben wird. *smail* akzeptiert auch numerische (Dotted Quad) Adressen als gültige Hostnamen, solange diese über einen *gethostbyaddr*-Aufruf aufgelöst werden können. So wäre etwa *scrooge@[149.76.12.4]* eine gültige, wenn auch höchst ungewöhnliche Mail-Adresse für **scrooge** auf **quark.physics.groucho.edu**.

  Wenn Ihr Rechner im Internet hängt, sind diese Router aber nicht das, was Sie suchen, weil sie MX-Records nicht unterstützen. Was in einem solchen Fall zu tun ist, können Sie im folgenden Abschnitt nachlesen.
- Wenn die pathalias-Datenbank */usr/lib/smail/paths* existiert, versucht *smail*, den Zielhost (ohne das anhängige **.uucp**) in dieser Datei zu finden. Mail an eine Adresse, die von diesem Router erkannt wird, wird über UUCP ausgeliefert, wobei der in der Datenbank gefundene Pfad verwendet wird.
- Die Hostadresse (ohne das angehängte **.uucp**) wird mit der Ausgabe des *uuname*-Befehls verglichen, um zu prüfen, ob der Zielhost wirklich ein UUCP-Nachbar ist. Ist das der Fall, wird die

Nachricht über UUCP ausgeliefert.

- Wird die Adresse von keinem der oberen Router erkannt, wird sie an den Smart Host weitergegeben. Der Pfad auf diesen Host sowie die zu verwendende Transportmethode sind in der *config*-Datei definiert.

Die Standardwerte funktionieren für viele einfache Setups, versagen aber, wenn die Anforderungen an das Routing komplizierter werden. Wenn Sie mit einem der nachfolgend beschriebenen Probleme konfrontiert werden, müssen Sie eine eigene *routers*-Datei installieren, die die Standardwerte überschreibt. Einige Linux-Distributionen werden mit einer Reihe von Konfigurations-Dateien geliefert, die bereits so geändert wurden, daß diese Probleme nicht auftreten. Schlagen Sie in Anhang B nach.

Die größten Probleme ergeben sich, wenn Ihr Host in einer Welt lebt, in der sowohl IP- als auch UUCP-Links gemeinsam existieren. Sie werden dann Hostnamen in Ihrer *hosts*-Datei besitzen, mit denen Sie sich nur gelegentlich über den SLIP-Link unterhalten. *smail* wird also versuchen, alle Post für diese Hosts über SMTP auszuliefern. Das ist aber normalerweise nicht das, was Sie wollen, weil selbst wenn der SLIP-Link regelmäßig aktiviert wird, SMTP wesentlich langsamer ist als UUCP. Mit dem Standard-Setup bleibt Ihnen aber keine andere Wahl.

Sie können dieses Problem umgehen, indem Sie dafür sorgen, daß *smail* die *paths*-Datei prüft, bevor es den Resolver befragt. Alle Hosts, bei denen Sie die Auslieferung mit UUCP erzwingen möchten, tragen Sie in die *paths*-Datei ein. Sollen *niemals* Nachrichten über SMTP übertragen werden, können Sie die Resolver-basierten Router auch ganz entfernen.

Ein weiteres Problem ist, daß das Standard-Setup kein echtes Internet-Mailrouting anbietet, weil der Resolver-basierte Router keine MX-Records verarbeiten kann. Um das Internet-Mailrouting vollständig zu unterstützen, müssen Sie den Standard-Router auskommentieren und dafür den aktivieren, der BIND verwendet. Allerdings existieren in einigen Linux-Distributionen *smail*-Binaries, die ohne BIND-Unterstützung kompiliert wurden. Wenn Sie BIND aktivieren und in Ihrer *paniclog*-Datei eine Meldung wie »`router inet_hosts: driver bind not found`« finden, müssen Sie sich die Sourcen besorgen und *smail* neu kompilieren (Informationen dazu finden Sie im obigen Abschnitt »LAN-Setup«).

Es ist übrigens keine gute Idee, den *uuname*-Treiber zu verwenden. Zum einen wird ein Konfigurations-Fehler generiert, wenn Sie UUCP nicht installiert haben, weil der *uuname*-Befehl dann nicht gefunden werden kann. Ein zweites Problem taucht auf, wenn Sie mehr Sites in Ihrer UUCP-*sys* eingetragen haben, als Mail-Links vorhanden sind. Das können Sites sein, mit denen Sie nur News austauschen, oder von denen Sie gelegentlich Dateien herunterladen, mit denen ansonsten aber keine weitere Kommunikation stattfindet.

Das erste Problem können Sie umgehen, indem Sie *uuname* durch ein Shell-Script ersetzen, das einfach nur *exit 0* ausführt. Die allgemeinere Lösung besteht aber darin, diese Treiber ganz aus der *routers*-Datei zu entfernen.

Die paths-Datenbank

smail erwartet, die pathalias-Datenbank in der Datei *paths* unter */usr/lib/smail* vorzufinden. Diese Datei ist optional, d. h. wenn Sie kein pathalias-Routing durchführen wollen, entfernen Sie einfach eine existierende *paths*-Datei.

Bei *paths* handelt es sich um eine sortierte ASCII-Datei, die aus Einträgen besteht, die Sitenamen auf UUCP-Bang-Pfade abbilden. Die Datei muß sortiert sein, weil *smail* Sites über einen binären Suchalgorithmus ermittelt. Kommentare sind in dieser Datei nicht erlaubt, und Sitenamen müssen vom Pfad durch einen Tabulator getrennt sein. pathalias-Datenbanken werden in Kapitel 13, *Elektronische Post* ausführlicher behandelt.

Wenn Sie diese Datei von Hand erzeugen, müssen Sie darauf achten, daß alle für eine Site gültigen Namen vorhanden sind. Ist eine Site beispielsweise sowohl über ihren reinen UUCP-Namen als auch über ihren voll qualifizierten Domainnamen bekannt, müssen Sie für jeden Namen einen Eintrag einfügen. Die Datei kann sortiert werden, indem Sie sie über eine Pipe durch den *sort*-Befehl laufen lassen.

Wenn sich Ihre Site nur am Ende des Baums befindet (d. h. eine Leaf-Site ist), sollte überhaupt keine *paths*-Datei notwendig sein. Setzen Sie einfach die Smart Host-Attribute in Ihrer *config*-Datei, und überlassen Sie das gesamte Routing dem Server, der Sie mit Mail versorgt.

Nachrichten an lokale Adressen ausliefern

Häufig ist die lokale Adresse einfach der Loginname eines Benutzers. In diesem Fall wird die Nachricht in die Mailbox des Benutzers (*/var/spool/mailBenutzername*) ausgeliefert. Andere Möglichkeiten sind Aliases und Namen von Mailing-Listen sowie Mail-Weiterleitung (Forwarding) des Benutzers. In solchen Fällen wird die lokale Adresse in eine Liste neuer Adressen umgewandelt, die lokal, aber auch auf einem anderen Rechner liegen können.

Neben diesen »normalen« Adressen kann *smail* auch mit anderen Arten lokaler Zieladressen wie Dateinamen und Pipe-Befehlen umgehen. Nun sind dies aber keine allgemein gültigen Adressen, d. h. Sie können nicht einfach eine Nachricht an */etc/passwd@vbrew.com* schicken. Sie sind nur gültig, wenn sie aus einer Weiterleitungs- oder Alias-Datei stammen.

Alles was mit einem Slash (/) oder einer Tilde (~) beginnt, wird als *Dateiname* interpretiert. Die Tilde verweist auf das Home-Verzeichnis des Benutzers und kann nur verwendet werden, wenn der Dateiname aus einer *.forward*-Datei oder aus einem Weiterleitungseintrag in der Mailbox (siehe unten) stammt. Wird eine Nachricht an eine Datei ausgeliefert, wird diese von *smail* an die Datei angehängt. Existiert die Datei noch nicht, wird sie automatisch erzeugt.

Bei einem *Pipe-Befehl* kann es sich um jeden UNIX-Befehl handeln, dem ein Pipe-Symbol (|) vorangestellt ist. *smail* übergibt diesen Befehl dann zusammen mit seinen Argumenten an die Shell, allerdings ohne das führende |. Die Nachricht selbst wird dem Befehl durch die Standardeingabe übergeben.

Um beispielsweise eine Mailing-Liste in eine lokale Newsgruppe einzuspeisen, könnten Sie ein Shell-Script namens *gateit* verwenden und ein lokales Alias einrichten, das alle Nachrichten aus dieser Mailing-Liste mit Hilfe von *»|gateit«* ausliefert.

Falls Leerzeichen in der Befehlszeile vorhanden sind, muß der Aufruf in Anführungszeichen stehen. Aus Sicherheitsgründen achtet smail darauf, daß der Befehl nicht ausgeführt wird, wenn die Adresse auf dubiose Weise entstanden ist (beispielsweise, wenn die Alias-Datei, aus der die Adresse stammt, von jedem geschrieben werden kann).

Lokale Benutzer

In den weitaus häufigsten Fällen bezeichnen lokale Adressen die Mailbox eines Benutzers. Die Mailbox ist in */var/spool/mail* zu finden und hat den Namen des jeweiligen Benutzers. Sie gehört dem Benutzer, die verwendete Gruppe ist mail, und der verwendete Modus ist 660. Falls sie nicht existiert, wird sie von *smail* automatisch erzeugt.

Beachten Sie, daß, obwohl */var/spool/mail* momentan der Standardort ist, an dem Mailbox-Dateien zu halten sind, manche Mail-Programme andere Pfade, z. B. */usr /spool/mail*, verwenden. Falls die Auslieferung an Benutzer Ihrer Maschine wiederholt fehlschlägt, sollten Sie versuchen, ob ein symbolischer Link auf */var /spool/mail* hilft.

Zwei Adressen werden von *smail* benötigt: **MAILER-DAEMON** und **postmaster**. Wird eine Bounce-Message erzeugt, weil eine Nachricht nicht zugestellt werden konnte, wird eine Kopie an den **postmaster**-Account zur Überprüfung geschickt (falls es sich um ein Konfigurations-Problem handeln sollte). **MAILER-DAEMON** wird als Absender einer Bounce-Message benutzt.

Besitzen diese Adressen keinen gültigen Account auf Ihrem System, bildet *smail* **MAILER-DAEMON** implizit auf **postmaster** ab, und **postmaster** auf **root**. Sie sollten dies aber überschreiben, indem Sie für **postmaster** ein Alias einrichten, das auf die Person verweist, die für die Administration der Mail-Software zuständig ist.

Weiterleitung (Forwarding)

Ein Benutzer kann seine Mail an eine andere Adresse umleiten, indem er eine der beiden von *smail* für diesen Zweck vorgesehenen Methoden verwendet. Eine Möglichkeit besteht darin,

```
Forward to Empfänger, …
```

in die erste Zeile der Mailbox-Datei einzutragen. Alle eingehenden Nachrichten werden dann an die in der Liste aufgeführten Empfänger verschickt. Alternativ können Sie in Ihrem Home-Verzeichnis eine *.forward*-Datei erzeugen, die die durch Kommata separierte Liste der Empfänger enthält. Bei dieser Variante werden alle Zeilen der Datei gelesen und interpretiert.

Es kann übrigens jede Art von Adresse verwendet werden. Ein nützliches Anwendungsbeispiel für eine *.forward*-Datei liefert uns die Urlaubszeit:

```
janet, "|vacation"
```

Die erste Adresse liefert die eingehende Nachricht wie gehabt an *janet*s Mailbox aus, während der *vacation*-Befehl einen kurzen Hinweis an den Absender schickt.

Alias-Dateien

smail kann mit Alias-Dateien umgehen, die mit denen von Berkeleys *sendmail* kompatibel sind. Einträge in einer Alias-Datei können die folgende Form haben:

```
 Alias: Empfänger
```

Empfänger ist eine durch Kommata separierte Liste von Adressen, durch die das Alias ersetzt wird. Die Empfängerliste kann sich über mehrere Zeilen hinziehen, wobei eine neue Zeile aber mit einem Tabulator beginnen muß.

Durch ein besonderes Feature ist *smail* auch in der Lage, Mailing-Listen aus der Alias-Datei zu verwalten: Wenn Sie als Empfänger `:include:`*`Dateiname`* angeben, liest *smail* die angegebene Datei und verwendet deren Inhalt als Liste der Empfänger.

Die zentrale Aliasdatei heißt */usr/lib/aliases*. Wenn Sie versuchen, diese Datei für jeden schreibbar zu machen, übergibt *smail* keine Nachrichten an Shell-Befehle, die eventuell in dieser Datei auftauchen. Nachfolgend ein Beispiel für eine *aliases*-Datei:

```
# Datei: vbrew.com /usr/lib/aliases
hostmaster: janet
postmaster: janet
usenet: phil
# die Entwickler-Mailingliste
development: joe, sue, mark, biff
        /var/mail/log/development
owner-development: joe
# Ankündigungen von allgemeinem Interesse werden
# an alle Mitarbeiter weitergeleitet
announce: :include: /usr/lib/smail/staff,
        /var/mail/log/announce
owner-announce: root
# PPP-Mailing-Liste in lokale Newsgruppe einspeisen
ppp-list: "|/usr/local/lib/gateit local.lists.ppp"
```

Tritt ein Fehler auf, während eine Nachricht an eine Adresse geschickt wird, die aus der *aliases*-Datei ermittelt wurde, schickt *smail* eine Kopie der Fehlermeldung an den »Besitzer des Alias«. Schlägt zum Beispiel die Auslieferung an *biff* fehl, während Nachrichten an die *development*-Mailing-Liste übertragen werden, geht eine Kopie der Fehlermeldung sowohl an den Absender als auch an *postmaster* und *owner-development*. Existiert keine Besitzeradresse, wird keine zusätzliche Fehlermeldung erzeugt.

Bei Auslieferungen an Dateien oder beim Aufruf von Programmen, die in der *aliases*-Datei stehen, wird *smail* zum Benutzer **nobody**, um Sicherheitsprobleme zu vermeiden. Das kann recht ärgerlich sein, besonders, wenn Nachrichten in Dateien weitergeleitet werden. In der obigen Datei beispielsweise müssen die Logdateien **nobody** gehören und von ihm beschreibbar sein, oder die Auslieferung schlägt fehl.

Mailing-Listen

Anstelle der *aliases*-Datei können Mailing-Listen auch über Dateien im Verzeichnis */usr/lib/smail/lists* verwaltet werden. Eine Mailing-Liste namens `nag-bugs` wird in der Datei *lists/nag-bugs* beschrieben, die eine durch Kommata separierte Liste der Teilnehmeradressen enthalten sollte. Die Liste kann sich über mehrere Zeilen erstrecken, Kommentare werden durch ein Doppelkreuz eingeleitet.

Für jede Mailing-Liste sollte ein Benutzer (oder Alias) namens **owner-*Listenname*** existieren. Alle

Fehler, die bei der Auflösung einer Adresse auftreten, werden diesem Benutzer gemeldet. Diese Adresse wird auch bei allen ausgehenden Nachrichten als Absenderadresse im Header-Feld `Sender:` verwendet.

UUCP-basierte Transportarten

Eine Reihe der in *smail* eingebundenen Transportarten verwenden das UUCP-Paket. In einer UUCP-Umgebung werden Nachrichten üblicherweise so weitergeleitet, daß *rmail* auf dem nächsten Host ausgeführt wird, wobei die Nachricht über die Standardeingabe und die Empfangsadresse in der Kommandozeile übergeben werden. Auf Ihrem Host sollte *rmail* ein Link auf den *smail*-Befehl sein.

Wird eine Nachricht an UUCP übergeben, wandelt *smail* die Zieladresse in einen UUCP-Bang-Pfad um. Beispielsweise wird *benutzer@host* in *host!benutzer* umgewandelt. Jedes Vorkommen des Adreß-Operators *%* bleibt erhalten, d. h. *benutzer%host@gateway* wird zu *gateway!benutzer%host*. Allerdings generiert *smail* solche Adressen niemals selbst.

Alternativ kann *smail* BSMTP-Batches über UUCP senden und empfangen. Bei BSMTP werden eine oder mehrere Nachrichten zu einem Paket verschnürt, das auch die Befehle enthält, die der lokale Mailer ausführen würde, wenn eine echte SMTP-Verbindung aufgebaut worden wäre. BSMTP wird häufig bei »Store-and-Forward-Netzwerken« (z. B. UUCP-basiert) verwendet, um Festplattenplatz zu sparen. Die *transports*-Beispieldatei im Anhang B enthält eine Transportart namens `bsmtp`, die einen Teil der BSMTP-Batches in einem Queue-Verzeichnis erzeugt. Die vollständigen Batches können dann später durch ein Shell-Script erzeugt werden, das die benötigten `HELO`- und `QUIT`-Befehle einfügt.

Um die Transportart `bsmtp` für bestimmte UUCP-Links zu aktivieren, müssen Sie mit sogenannten *method*-Dateien arbeiten. Wenn Sie nur einen UUCP-Link haben und den Smart Host-Router verwenden, können Sie das Senden von SMTP-Batches aktivieren, indem Sie die Konfigurations-Variable `smart_transport` auf `bsmtp` anstelle von `uux` setzen.

Um SMTP-Batches über UUCP empfangen zu können, müssen Sie sicherstellen, daß sich der Befehl auf Ihrem Rechner befindet, an den die andere Seite die Pakete schickt, um sie zu entpacken. Arbeitet die andere Seite ebenfalls mit *smail*, müssen Sie *rsmtp* als Link auf *smail* definieren. Verwendet die Gegenseite *sendmail*, müssen Sie zusätzlich noch ein Script namens */usr/bin/bsmtp* erzeugen, das einfach *exec rsmtp* ausführt (ein symbolischer Link würde nicht funktionieren).

SMTP-basierte Transportarten

smail unterstützt momentan einen SMTP-Treiber, mit dem Mail über TCP-Verbindungen ausgeliefert werden kann.(5) Es ist in der Lage, Nachrichten an eine beliebige Anzahl von Adressen auf einem einzelnen Host auszuliefern. Der Hostname kann dabei entweder ein voll qualifizierter Domainname sein, der von der Netzwerk-Software aufgelöst werden kann, oder ein Wert in Dotted Quad Notation, der von eckigen Klammern umschlossen ist. Üblicherweise werden alle Adressen, die von einem der BIND-, *gethostbyname*- oder *gethostbyaddr*-Router-Treiber aufgelöst werden, an SMTP weitergeleitet.

Der SMTP-Treiber versucht direkt über den in */etc/services* eingetragenen `smtp`-Port eine Verbindung zum entfernten Host aufzubauen. Kann dieser nicht erreicht werden, oder kommt es zu einem Timeout, wird die Auslieferung zu einem späteren Zeitpunkt wiederholt.

Für die Auslieferung im Internet werden die Routen zum Zielhost in dem im Kapitel 13 beschriebenen *route-addr*-Format benötigt. Ein UUCP-Bang-Pfad ist hier nicht zu gebrauchen.(6) Darum wandelt *smail* die Adresse *benutzer%host@gateway*, wobei **gateway** über *host1!host2!host3* erreicht werden kann, in die Quellrouten-Adresse *<@host2,@host3:benutzer%host@gateway>* um, die dann als Zieladresse an *host1* übergeben wird. Um diese Umwandlungen (zusammen mit dem eingebauten BIND-Treiber) zu aktivieren, müssen Sie den Eintrag für den `smtp`-Treiber in der *transports*-Datei editieren. Mehr dazu in Anhang B.

Qualifizierung von Hostnamen

Manchmal ist es wünschenswert, unqualifizierte Hostnamen (z. B. ohne einen Domainnamen) abzufangen, die in Adressen für Absender oder Empfänger auftauchen. Ein Beispiel ist ein Gateway zwischen zwei Netzwerken, von denen eines voll qualifizierte Domainnamen benötigt. Bei einem Internet/UUCP-Relay sollten unqualifizierte Hostnamen standardmäßig auf die Domain **uucp** abgebildet werden. Weitergehende Adreß-Modifikationen sind fragwürdig.

Die Datei */usr/lib/smail/qualify* teilt *smail* mit, welche Domainnamen an welche Hostnamen anzuhängen sind. Einträge in der *qualify*-Datei beginnen mit einem Hostnamen in der ersten Spalte, dem der Domainname folgt. Zeilen, die mit einem Doppelkreuz beginnen (wobei führende Leerzeichen, Tabulatoren etc. ignoriert werden), werden als Kommentare betrachtet. Die Einträge werden linear Zeile für Zeile durchsucht.

Existiert keine *qualify*-Datei, wird auch keine Qualifikation durchgeführt.

Der spezielle Hostname * steht für jeden Host. Auf diese Weise können Sie alle bisher noch nicht aufgeführten Hosts in einer Standard-Domain zusammenfassen. Diese Zeile sollte immer der letzte Eintrag in Ihrer Datei sein.

Bei der virtuellen Brauerei sind alle Hosts so eingerichtet, daß voll qualifizierte Domainnamen in den Absenderadressen verwendet werden. Bei unqualifizierten Empfangsadressen wird die **uucp**-Domain angenommen, so daß nur ein einzelner Eintrag in der *qualify*-Datei benötigt wird:

```
# /usr/lib/smail/qualify, zuletzt geändert am 12.02.1994 von janet
#
*             uucp
```

Fußnoten

(1)

Entsprechend dem Linux-Dateisystem-Standard der neue Standardplatz für sendmail. Ein anderer gängiger Ort ist/usr/lib.

(2)

Dafür gibt es folgenden Grund: Stellen Sie sich vor, daß Ihr Hostname monad lautet, aber nicht in den Maps registriert ist. Nun existiert in den Maps aber ein Rechner namens monad. Alle E-Mails an monad!root werden dann an diese Site weitergeleitet, selbst wenn sie von einem Ihrer direkten

UUCP-Nachbarn stammen.

(3)

Haben Sie smail mit der Linux-Distribution eines Anbieters bezogen, dann haben Sie entsprechend den Kopierbestimmungen von smail »gegen eine nominelle Versandgebühr« auch Anspruch auf den Quellkode.

(4)

Die Standard-Konfigurationsdateien befinden sich in samples/generic unterhalb des Source-Verzeichnisses.

(5)

Die Autoren nennen die Unterstützung »einfach«. Für zukünftige Versionen von smail haben sie ein komplettes Backend angekündigt, das diese Arbeiten effektiver erledigen wird.

(6)

Allerdings wird die Verwendung von Routen im Internet nicht empfohlen. Sie sollten statt dessen voll qualifizierte Domainnamen verwenden.

Kapitel 15
Sendmail+IDA

Eine Einführung in Sendmail+IDA

Es wird behauptet, daß man kein *echter* UNIX-Systemadministrator sei, solange man noch keine *sendmail.cf*-Datei editiert hat. Es wird auch behauptet, daß man verrückt sei, wenn man es ein zweites Mal tut.

sendmail ist ein unglaublich mächtiges Programm. Für die meisten Leute ist es aber auch unglaublich schwer zu erlernen und zu verstehen. Jedes Programm, dessen definitive Referenz 792 Seiten lang ist, schreckt die meisten Leute zu Recht ab. Sendmail+IDA ist anders. Es befreit Sie davon, die ewig kryptische *sendmail.cf*-Datei bearbeiten zu müssen, und erlaubt es dem Administrator, die Site-spezifische Routing- und Adreß-Konfiguration über relativ einfache Support-Dateien, sogenannte »Tabellen« (*tables*), zu erledigen. Ein Wechsel zu Sendmail+IDA kann Ihnen viele Stunden Arbeit und Streß ersparen.

Verglichen mit den anderen großen Mail-Transport-Agents gibt es nichts, was mit Sendmail+IDA nicht schneller und einfacher erledigt werden könnte. Typische Aufgaben, die auf einer normalen UUCP- oder Internet-Site anfallen, sind einfach zu lösen. Normalerweise nur sehr schwer durchzuführende Konfigurationen sind einfach zu erzeugen und zu verwalten.

Während dieses Buch entsteht, ist die aktuelle Version, *sendmail5.67b+ IDA1.5*, über FTP bei **vixen.cso.uiuc.edu** zu beziehen. Unter Linux kann das Programm ohne jegliche Anpassungen kompiliert werden.

Alle Konfigurations-Dateien, die benötigt werden, um die Sendmail+IDA-Quellen zu kompilieren, zu installieren und unter Linux zum Laufen zu bringen, sind in *newspak-2.2.tar.gz* enthalten, das Sie über FTP von **sunsite.unc.edu** aus dem Verzeichnis */pub/Linux/system/Mail* herunterladen können.

Übersicht der Konfigurations-Dateien

Das Setup von *sendmail* erfolgt traditionell durch eine System-Konfigurationsdatei (typischerweise */etc/sendmail.cf* oder */usr/lib/sendmail.cf*, die an keine Sprache erinnert, die Sie bisher kennengelernt haben. Das Editieren von *sendmail.cf* zur Anpassung des Verhaltens kann eine demütigende Erfahrung sein.

Dank Sendmail+IDA gehören diese Qualen aber der Vergangenheit an, weil alle Konfigurations-Optionen über Tabellen gesteuert werden, die eine ziemlich einfach zu verstehende Syntax verwenden. Diese Optionen werden konfiguriert, indem *m4* (ein Makroprozessor) oder *dbm* (ein Datenbankprozessor) über Make-Files auf eine Reihe von Datendateien angewandt werden, die den Quellen beiliegen.

Die Datei *sendmail.cf* definiert nur das Standardverhalten des Systems. Nahezu jede spezielle Anpassung wird über eine Reihe optionaler Tabellen durchgeführt. *sendmail.cf* selbst muß dazu nicht editiert werden. Die nachfolgende Liste beschreibt die *sendmail*-Tabellen:

mailertable
: Definiert spezielle Verhaltensweisen für andere Hosts oder Domains.

uucpxtable
: Erzwingt die UUCP-Auslieferung von Nachrichten an Hosts, die das DNS-Format verwenden.

pathtable
: Definiert UUCP-Bang-Pfade zu anderen Hosts und Domains.

uucprelays
: Kürzt den pathalias-Pfad für wohlbekannte entfernte Hosts ab.

genericfrom
: Wandelt interne in generische Adressen um, die für die Außenwelt sichtbar sind.

xaliases
: Wandelt generische Adressen in/von gültige(n) interne(n) um.

decnetxtable
: Wandelt RFC-822-Adressen in DECnet-Adressen um.

Die Datei sendmail.cf

sendmail.cf wird für Sendmail+IDA nicht direkt bearbeitet, sondern durch eine *m4*-Konfigurations-Datei erzeugt, die vom lokalen Systemadministrator bereitgestellt wird. Diese Datei bezeichnen wir nachfolgend als *sendmail.m4*.

Die Datei enthält einige Definitionen und zeigt ansonsten nur auf die Tabellen, wo die eigentliche Arbeit erledigt wird. Im allgemeinen müssen die folgenden Informationen spezifiziert werden:

- Die auf dem lokalen System verwendeten Pfad- und Dateinamen
- Der oder die Namen, unter denen diese Site für E-Mail-Zwecke bekannt ist
- Welcher Standard-Mailer (und vielleicht welcher Smart Host) gewünscht wird

Eine Reihe von Parametern kann definiert werden, mit denen das Verhalten des lokalen Servers bestimmt wird oder vorkompilierte Konfigurations-Optionen überschrieben werden können. Die Konfigurations-Optionen sind in der Datei *ida/cf/OPTIONS* im Source-Verzeichnis zu finden.

Eine *sendmail.m4*-Datei für eine Minimalkonfiguration (UUCP oder SMTP, wobei alle nichtlokalen Mails an einen direkt verbundenen Smart Host weitergeleitet werden) kann aus nur 10 bis 15 Befehlen (ohne Kommentarzeilen) bestehen.

Eine sendmail.m4-Beispieldatei

Eine *sendmail.m4*-Datei für vstout in der virtuellen Brauerei ist in Beispiel 15--1 zu sehen. vstout verwendet SMTP zur Kommunikation mit allen Hosts im Brauerei-LAN und sendet alle Mails für andere Zielorte über UCCP an moria, seinen Internet-Verteilerhost.

Tatsächlich nennen die meisten Leute ihre Konfigurations-Datei nicht *sendmail.m4*, sondern benennen sie nach dem Host, in unserem Beispiel also *vstout.m4*. Der Name spielt aber keine Rolle, solange die Ausgabedatei *sendmail.cf* genannt wird.

Beispiel 15-1. vstout.m4-Konfigurations-Datei (Beispiel)

```
dnl #------------------ SENDMAIL.M4-BEISPIELDATEI------------------
dnl # (der String 'dnl' ist das m4-Equivalent für das Auskommentieren einer Zeile)
dnl # im allgemeinen wird LIBDIR aus den vorkompilierten Pfaden nicht überschrieben
dnl #define(LIBDIR,/usr/local/lib/mail)dnl    # hier sind alle Support-Dateien zu
finden
define(LOCAL_MAILER_DEF, mailers.linux)dnl    # Mailer für lokale Auslieferung
define(POSTMASTERBOUNCE)dnl                   # postmaster bekommt gebouncte
```

```
Nachrichten
define(PSEUDODOMAINS, BITNET UUCP)dnl         # DNS nicht auf diese anwenden
dnl #--------------------------------------------------------------
dnl #
define(PSEUDONYMS, vstout.vbrew.com  vstout.UUCP vbrew.com)
dnl                                           # Namen, unter denen wir bekannt sind
define(DEFAULT_HOST, vstout.vbrew.com)dnl     # unser primärer 'Name' für Mail
define(UUCPNAME, vstout)dnl                   # unser UUCP-Name
dnl #
dnl #--------------------------------------------------------------
dnl #
define(UUCPNODES, |uuname|sort|uniq)dnl       # unsere UUCP-Nachbarn
define(BANGIMPLIESUUCP)dnl                    # sicherstellen, daß UUCP-Mail
define(BANGONLYUUCP)dnl                       #  korrekt gehandhabt wird
define(RELAY_HOST, moria)dnl                  # unser smarter Verteilerhost
define(RELAY_MAILER, UUCP-A)dnl               # moria wird über UUCP erreicht
dnl #
dnl #---------------------------------------------------------------------
dnl #
dnl # Verschiedene dbm-Suchtabellen
dnl #
define(ALIASES, LIBDIR/aliases)dnl            # System-Aliases
define(DOMAINTABLE, LIBDIR/domaintable)dnl    # Hosts Domains zuordnen
define(PATHTABLE, LIBDIR/pathtable)dnl        # Pfad-Datenbank
define(GENERICFROM, LIBDIR/generics)dnl       # Interne in generische Adressen
define(MAILERTABLE, LIBDIR/mailertable)dnl    # Mailer je Host oder Domain
define(UUCPXTABLE, LIBDIR/uucpxtable)dnl      # Pfade zu von uns versorgten Hosts
define(UUCPRELAYS, LIBDIR/uucprelays)dnl      # Kurze Pfade für bekannte Hosts
dnl #
dnl #---------------------------------------------------------------------
dnl #
dnl # füge hier den "echten" Kode ein, der alles zum Laufen bringt
dnl # (ist im Source-Kode enthalten)
dnl #
include(Sendmail.mc)dnl                         # BENÖTIGTER EINTRAG!!!
dnl #
dnl #------------ ENDE DER SENDMAIL.M4-BEISPIELDATEI -------
```

Anwendungsspezifische sendmail.m4-Parameter

Einige Punkte in der Datei *sendmail.m4* werden die ganze Zeit über benötigt, während andere ignoriert werden können, wenn Ihnen die Standardeinstellungen genügen. Die folgenden Abschnitte behandeln detailliert jeden der Punkte aus der *sendmail.m4*-Datei aus Beispiel 15--1.

Punkte, die Pfade definieren

```
dnl #define(LIBDIR,/usr/local/lib/mail)dnl  # hier sind alle Support-Dateien
                                            # zu finden
```

`LIBDIR` definiert das Verzeichnis, in dem Sendmail+IDA alle Konfigurations-Dateien, die verschiedenen *dbm*-Tabellen und spezielle lokale Definitionen erwartet. Bei einer typischen Binär-Distribution ist dieser Pfad bereits in die *sendmail*-Binary kompiliert und muß in *sendmail.m4* nicht explizit gesetzt werden.

Die obige Zeile beginnt mit `dnl`, was bedeutet, daß es sich bei der Zeile um einen Kommentar handelt.

Sollen die Support-Dateien an einer anderen Stelle gespeichert werden, müssen Sie das führende `dnl` aus der obigen Zeile entfernen, den gewünschten Pfad eintragen und *sendmail.cf* neu erzeugen und installieren.

Definieren des lokalen Mailers

```
define(LOCAL_MAILER_DEF, mailers.linux)dnl  # Mailer für lokale Auslieferung
```

Die meisten Betriebssysteme bieten ein Programm an, das die Auslieferung lokaler Mails übernimmt. Typische Programme für viele der Hauptvarianten von UNIX sind bereits in das *sendmail*-Binary integriert.

Unter Linux muß der lokale Mailer explizit definiert werden, weil ein Programm zur lokalen Auslieferung nicht zwangsläufig in Ihrer Distribution vorhanden sein muß. Dies wird durch die Spezifizierung von `LOCAL_MAILER_DEF` in *sendmail.m4* erreicht.

Um diesen Service beispielsweise durch das häufig verwendete Programm *deliver* ausführen zu lassen, würden Sie `LOCAL_MAILER_DEF` auf *mailers.linux* setzen.(1)

Die folgende Datei sollte dann als *mailers.linux* in dem unter `LIBDIR` angegebenen Verzeichnis installiert werden. Hier wird *deliver* explizit als lokaler Mailer in `Mlocal` definiert, wobei gleichzeitig die richtigen Parameter eingetragen werden, so daß *sendmail* alle Nachrichten korrekt ausliefern kann, die für das lokale System bestimmt sind. Solange Sie kein *sendmail*-Experte sind, sollten Sie das folgende Beispiel nicht verändern.

```
# -- /usr/local/lib/mail/mailers.linux --
#     (Lokale Mailer für Linux )
Mlocal, P=/usr/bin/deliver, F=SlsmFDMP, S=10, R=25/10, A=deliver $u
Mprog,  P=/bin/sh,       F=lsDFMeuP,   S=10, R=10, A=sh -c $u
```

Es gibt auch einen eingebauten Standardwert für *deliver* in der Datei *Sendmail.mc*, die in *sendmail.cf* eingefügt wird. Soll dieser verwendet werden, würden Sie nicht die Datei *mailers.linux* verwenden, sondern würden folgende Zeilen in Ihre *sendmail.m4* eintragen:

```
dnl  --  (in sendmail.m4)  -
define(LOCAL_MAILER_DEF, DELIVER)dnl       # Mailer für lokale Auslieferung
```

Linux Unglücklicherweise setzt *Sendmail.mc* voraus, daß *deliver* in */bin* installiert ist. Das ist bei Slackware1.1.1 nicht der Fall (hier wird es in */usr/bin* installiert). In diesem Fall müßten Sie entweder einen entsprechenden Link einrichten, oder *deliver* erneut aus den Quellen erzeugen, so daß es dann in */bin* zu finden ist.

Mit gebouncter Mail umgehen

```
define(POSTMASTERBOUNCE)dnl        # postmaster bekommt gebouncte Nachrichten
```

Für viele Sites ist es wichtig, sicherzustellen, daß Nachrichten mit nahezu 100prozentiger Sicherheit verschickt und empfangen werden. Obwohl es hilfreich ist, die *syslogd*-Logs zu untersuchen, muß sich der Mail-Administrator auch die Header der nicht zugestellten Nachrichten ansehen, um erkennen zu können, ob es sich um einen Fehler des Benutzers oder um einen Konfigurations-Fehler bei einem der beteiligten Systeme handelt.

Durch Definition von `POSTMASTERBOUNCE` wird eine Kopie jeder nicht zustellbaren Nachricht an die Person geschickt, die für dieses System als **Postmaster** definiert wurde.

Leider wird durch Einstellen dieses Parameters auch der *Inhalt* der Nachricht mit an den Postmaster übertragen, was möglicherweise Fragen über die Privatsphäre Ihrer Benutzer aufwirft.

Postmaster sollten sich grundsätzlich dazu anhalten, die Post fremder Leute nicht zu lesen (oder die technischen Möglichkeiten wie Shell-Scripten ausschöpfen und den Text von nicht ausgelieferten Nachrichten abschneiden).

DNS-bezogene Punkte

```
define(PSEUDODOMAINS, BITNET UUCP)dnl       # DNS nicht auf diesen anwenden
```

Es gibt verschiedene wohlbekannte Netzwerke, die in Mail-Adressen aus historischen Gründen immer wieder adressiert werden, die aber für DNS nicht gültig sind. Durch die Definition von `PSEUDODOMAINS` verhindern Sie sinnlose DNS-Lookups, die immer fehlschlagen würden.

Definition lokaler Systemnamen

```
define(PSEUDONYMS, vstout.vbrew.com  vstout.UUCP vbrew.com)
dnl                                        # Namen, unter denen wir bekannt sind
define(DEFAULT_HOST, vstout.vbrew.com)dnl  # unser primärer 'Name' für Mail
```

Häufig möchten Systeme ihre wahre Identität verbergen, als Mail-Gateways dienen oder Nachrichten empfangen und verarbeiten, die an »alte« Namen adressiert sind.

`PSEUDONYMS` spezifiziert eine Liste von Hostnamen, für die das lokale System Nachrichten akzeptiert.

`DEFAULT_HOST` spezifiziert den Hostnamen, der in Nachrichten erscheint, die vom lokalen Host verschickt wurden. Es ist wichtig, daß dieser Parameter auf einen gültigen Wert eingestellt wird, weil sonst keine zurückkommende Mail ausgeliefert werden könnte.

UUCP-bezogene Punkte

```
define(UUCPNAME, vstout)dnl                   # unser UUCP-Name
define(UUCPNODES, |uuname|sort|uniq)dnl       # unsere UUCP-Nachbarn
define(BANGIMPLIESUUCP)dnl                    # sicherstellen, daß UUCP-Mail
define(BANGONLYUUCP)dnl                       #  korrekt gehandhabt wird
```

Häufig sind Systeme für DNS-Zwecke unter dem einen und für UUCP-Zwecke unter einem anderen Namen bekannt. Mit `UUCPNAME` können Sie einen anderen Hostnamen definieren, der in den Headern aller ausgehenden UUCP-Mails erscheint.

`UUCPNODES` definiert die Befehle, die eine Liste der Hostnamen aller Systeme zurückliefern, mit denen Sie direkt über UUCP verbunden sind.

`BANGIMPLIESUUCP` und `BANGONLYUUCP` stellen sicher, daß Nachrichten, die in UUCP-Bang-Syntax adressiert wurden, entsprechend den UUCP-Richtlinien behandelt werden und nicht entsprechend den heute im Internet verwendeten DNS-Richtlinien.

Verteilerstationen und Mailer

```
define(RELAY_HOST, moria)dnl                  # unser smarter Verteilerhost
define(RELAY_MAILER, UUCP-A)dnl               # moria wird über UUCP erreicht
```

Viele Systemadministratoren wollen mit der ganzen Arbeit nicht belästigt werden, die notwendig ist, um sicherzustellen, daß ihr System alle Netzwerke und Systeme in allen Netzwerken weltweit erreichen kann. Statt dessen leiten sie alle Nachrichten an ein anderes System weiter, das als »smart« bekannt ist.

`RELAY_HOST` definiert den UUCP-Hostnamen eines solchen benachbarten smarten Systems.

`RELAY_MAILER` definiert, welcher Mailer bei der Weiterleitung an dieses System zu verwenden ist.

Es ist wichtig zu bedenken, daß ein Einstellen dieses Parameters bewirkt, daß alle ausgehenden Mails an dieses andere System weitergeleitet werden, was diesen Rechner unter Umständen erheblich belastet. Holen Sie sich also zuerst das Einverständnis des Postmasters des entfernten Host ein, bevor Sie Ihr System so konfigurieren, daß dieser als allgemeiner

Vermittlungshost benutzt wird.

Konfigurations-Tabellen

```
define(ALIASES, LIBDIR/aliases)dnl           # System-Aliases
define(DOMAINTABLE, LIBDIR/domaintable)dnl   # Hosts Domains zuordnen
define(PATHTABLE, LIBDIR/pathtable)dnl       # Pfad-Datenbank
define(GENERICFROM, LIBDIR/generics)dnl      # interne in generische Adressen
define(MAILERTABLE, LIBDIR/mailertable)dnl   # Mailer je Host oder Domain
define(UUCPXTABLE, LIBDIR/uucpxtable)dnl     # Pfade zu von uns versorgten Hosts
define(UUCPRELAYS, LIBDIR/uucprelays)dnl     # kurze Pfade für bekannte Hosts
```

Mit diesen Makros können Sie die Orte verändern, an denen Sendmail+IDA die verschiedenen *dbm*-Tabellen sucht, die das »wirkliche« Verhalten des Systems bestimmen. Es ist grundsätzlich eine weise Entscheidung, sie in `LIBDIR` zu belassen.

Die Hauptdatei Sendmail.mc

```
include(Sendmail.mc)dnl                         # BENÖTIGTER EINTRAG!!!
```

Die Autoren von Sendmail+IDA liefern die Datei *Sendmail.mc* mit, die die wirklich wichtigen Teile enthält, aus denen dann *sendmail.cf* erzeugt wird. Regelmäßig werden neue Versionen veröffentlicht, in denen Fehler behoben oder in die zusätzliche Funktionen aufgenommen wurden, ohne daß gleich eine völlig neue Release und eine Neukompilierung der *sendmail*-Quellen vonnöten wäre. Es ist wichtig, daß Sie die Datei *nicht* editieren.

Welche Einträge werden denn nun wirklich benötigt?

Wird keine der optionalen *dbm*-Tabellen verwendet, liefert Sendmail+IDA Mail über `DEFAULT_MAILER` (und eventuell über `RELAY_HOST` und `RELAY_MAILER`) aus, der in *sendmail.m4* definiert ist, und aus dem die eigentliche *sendmail.cf* generiert wird. Sie können dieses Verhalten einfach verändern, indem Sie entsprechende Einträge in *domaintable* oder *uucpxtable* vornehmen.

Eine typische Internet-Site, die DNS versteht, oder eine reine UUCP-Site, die alle Nachrichten über UUCP an einen smarten `RELAY_HOST` weiterleitet, benötigt wahrscheinlich überhaupt keine spezifischen Tabelleneinträge.

Nahezu jedes System sollte aber die `DEFAULT_HOST`- und `PSEUDONYMS`-Makros setzen, die den kanonischen Sitenamen und eventuelle Aliases definieren, unter denen es bekannt ist. Auch `DEFAULT_MAILER` sollte definiert sein. Wenn Sie nur einen Verteilerhost und -mailer betreiben, brauchen Sie diese Standardwerte nicht einzustellen, weil dies automatisch geschieht.

Bei UUCP-Hosts muß wahrscheinlich `UUCPNAME` auf den offiziellen UUCP-Namen gesetzt werden. Möglicherweise müssen auch `RELAY_MAILER` und `RELAY_HOST` eingestellt werden, mit denen das Smart Host-Routing über einen Mail-Verteiler aktiviert wird. Die für Mails zu verwendende Transportart wird in `RELAY_MAILER` definiert und sollte bei UUCP-Sites normalerweise *UUCP-A* enthalten.

Wenn Ihre Site ausschließlich SMTP verwendet und DNS spricht, müssen Sie `DEFAULT_MAILER` auf `TCP-A` einstellen und wahrscheinlich die Zeilen `RELAY_MAILER` und `RELAY_HOST` entfernen.

Die Sendmail+IDA-Tabellen

Sendmail+IDA besitzt eine Reihe von Tabellen, mit denen Sie die Standardarbeitsweise von *sendmail* (wie in *sendmail.m4* spezifiziert) überschreiben können. Gleichzeitig können Sie spezielle Verhaltensweisen für einzigartige Situationen, andere Systeme und Netzwerke festlegen. Diese Tabellen werden mit *dbm* über ein Makefile nachgearbeitet, das mit der Distribution geliefert wird.

Die meisten Sites benötigen, wenn überhaupt, nur wenige dieser Tabellen. Wenn Ihre Site diese Tabellen nicht braucht,

besteht die einfachste Möglichkeit darin, die Länge der Dateien auf Null Byte zu bringen (mit dem Befehl *touch*), und in `LIBDIR` das Standard-Makefile zu verwenden, anstatt das Makefile selbst zu editieren.

mailertable

mailertable definiert die spezielle Behandlung von spezifischen Hosts oder Domains, basierend auf dem Namen des entfernten Host oder des Netzwerks. Sie wird häufig bei Internet-Sites verwendet, um einen Mail-Verteilerhost oder ein Gateway anzusprechen, über den/das ein entferntes Netzwerk erreicht werden soll. Außerdem wird bestimmt, welches Protokoll (UUCP oder SMTP) dabei verwendet werden soll. UUCP-Sites benötigen diese Datei nicht.

Die Reihenfolge ist wichtig. *sendmail* liest die Datei von oben nach unten und verarbeitet die Nachricht entsprechend dem ersten passenden Eintrag. Es ist daher sinnvoll, die exaktesten Einträge direkt am Anfang einzugeben und die allgemeineren Einträge gegen Ende.

Stellen Sie sich vor, daß Sie alle Nachrichten an das Institut für Informatik an der Groucho-Marx-Universität über UUCP an den Verteilerhost **ada** weiterleiten wollen. Dazu müssen Sie in *mailertable* einen Eintrag wie den folgenden aufnehmen:

```
# (in mailertable)
#
# alle Nachrichten für die Domain .cs.groucho.edu über UUCP an ada
UUCP-A,ada         .cs.groucho.edu
```

Stellen Sie sich nun vor, daß alle Nachrichten an die größere Domain groucho.edu an einen anderen Verteilerhost namens **bighub** gehen sollen, der die Auflösung der Adressen und die Auslieferung übernehmen soll. *mailertable* müßte dafür wie folgt erweitert werden:

```
# (in mailertable)
#
# alle Nachrichten für die Domain .cs.groucho.edu über UUCP an ada
UUCP-A,ada         .cs.groucho.edu
#
# alle Nachrichten für die Domain groucho.edu über UUCP an bighub
UUCP-A,bighub      .groucho.edu
```

Wie bereits erwähnt, ist die Reihenfolge wichtig. Würden Sie die Reihenfolge dieser beiden Regeln vertauschen, würden alle Nachrichten an **.cs.groucho.edu** über den allgemeineren **bighub**-Pfad laufen und nicht über den eigentlich gewünschten **ada**-Pfad.

In den obigen *mailertable*-Beispielen läßt der *UUCP-A*-Mailer *sendmail* über UUCP ausliefern, wobei im Mail-Header Domain-Namen/voll qualifizierte Host-Namen verwendet werden.

Das Komma zwischen dem Mailer und dem entfernten System teilt ihm mit, die Nachricht zur Adreßauflösung und Auslieferung an **ada** weiterzuleiten.

mailertable-Einträge haben das folgende Format:

```
Mailer Trennzeichen Vermittlungshost Host_oder_Domain
```

Es gibt eine ganze Reihe möglicher Mailer. Sie unterscheiden sich hauptsächlich in der Art, wie Adressen behandelt werden. Typische Mailer sind *TCP-A* (TCP/IP mit Internet-Adressen), *TCP-U* (TCP/IP mit UUCP-Adressen) und *UUCP-A* (UUCP mit Internet-Adressen).

Das Zeichen, das den Mailer vom Hostteil links in *mailertable* trennt, bestimmt, wie die Adresse von *mailertable* modifiziert wird:

`!`

: Bei einem Ausrufezeichen wird der Hostname des Empfängers entfernt, bevor eine Weiterleitung an den Mailer erfolgt. Sie können dies verwenden, um die Auslieferung von Mail an eine fehlerhaft konfigurierte Site zu erzwingen.

,

: Ein Komma verändert eine Adresse nicht. Die Nachricht wird einfach über den angegebenen Mailer an den angegebenen Verteilerhost weitergeleitet.

:

: Bei einem Doppelpunkt wird der Hostname des Empfängers nur entfernt, wenn zwischen Ihnen und dem Zielhost noch weitere Hosts liegen. Das bedeutet also, daß *foo* aus *foo!bar!joe* entfernt wird, während *xyzzy!janet* unverändert bleibt.

Eine wichtige Sache ist, daß *mailertable* nur den Envelope neu schreibt (um die Nachricht auf das andere System zu bekommen). Andere Teile der Nachricht zu verändern, gilt als unfein, weil mit hoher Wahrscheinlichkeit die Mail-Konfiguration dabei zerbricht.

uucpxtable

Normalerweise werden Nachrichten an Hosts mit voll qualifiziertem Domainnamen über SMTP mittels DNS oder über den Verteilerhost ausgeliefert. *uucpxtable* erzwingt die Auslieferung über UUCP-Routing durch Umwandlung des mit der Domain versehenen Namens in den UUCP-typischen entfernten Hostnamen ohne Domain.

uucpxtable wird häufig beim Betrieb eines Mail-Forwarders für eine Site oder Domain eingesetzt, oder wenn ein direkter und zuverlässiger UUCP-Link anstelle des Standard-Mailers verwendet werden soll, um mehrere Sprünge (Hops) über dazwischenliegende Systeme und Netzwerke zu vermeiden.

UUCP-Sites, die mit UUCP-Nachbarn kommunizieren, bei denen Mailheader mit Domainnamen verwendet werden, würden diese Datei nutzen, um die Auslieferung der Nachrichten über die direkte UUCP-Punkt-zu-Punkt-Verbindung zwischen diesen beiden Systemen zu erzwingen, statt die weniger direkte Route über `RELAY_MAILER` und `RELAY_HOST` oder über `DEFAULT_MAILER` zu gehen.

Internet-Sites, die nicht über UUCP kommunizieren, werden *uucpxtable* wahrscheinlich nicht benutzen.

Stellen Sie sich vor, daß Sie das Mail-Forwarding für ein System bereitstellen, das unter DNS den Namen **sesame.com** besitzt und in den UUCP-Maps als **sesame** geführt wird. Sie müssen dann den folgenden Eintrag in Ihre *uucpxtable* aufnehmen, damit die Nachrichten für diesen Host über die direkte UUCP-Verbindung geleitet werden:

```
#============== /usr/local/lib/mail/uucpxtable ============
# Nachrichten an joe@sesame.com werden zu sesame!joe umgeschrieben
# und daher über UUCP ausgeliefert
#
sesame     sesame.com
#
#-----------------------------------------------------------
```

pathtable

pathtable wird verwendet, um Routen zu entfernten Hosts oder Netzwerken explizit zu definieren. *pathtable* verwendet die gleiche Syntax wie pathalias und sollte alphabetisch sortiert sein. Die beiden in einer Zeile auftretenden Felder sollten durch einen echten Tabulator voneinander getrennt sein, da sich *dbm* ansonsten beschweren könnte.

Die meisten Systeme benötigen keine *pathtable*-Einträge.

```
#=============== /usr/local/lib/mail/pathtable ================
#
# dies ist eine Pfaddatei, die eine pathalias-Syntax verwendet und es
# Ihnen erlaubt, Nachrichten über den direkten UUCP-Pfad zu Ihren
# UUCP-Nachbarn zu transportieren, so daß diese Nachrichten nicht erst den
# langen Weg über Ihren Smart Host nehmen müssen.
```

```
#
# verwenden Sie echte Tabulatoren, sonst könnte sich m4 beschweren
#
# Mails werden über eine oder mehrere dazwischenliegende Sites an
# ein entferntes System geschickt; dabei wird UUCP-Adressierung verwendet
#
sesame!ernie!%s                          ernie
#
# Weiterleitung zu einem System, das der UUCP-Nachbar einer erreichbaren
# Internet-Site ist
#
swim!%s                                     swim
#
# nachfolgend werden alle Nachrichten für zwei Netzwerke über zwei
# verschiedene Gateways verschickt (sehen Sie den führenden Punkt ?)
# in diesem Beispiel sind "uugate" und "byte" spezifische Systeme, die als
# Mail-Gateways für die Pseudo-Domains .UUCP bzw. .BITNET dienen
#
%@nngate.groucho.edu      .UUCP
byte!%s@mail.shift.com           .BITNET
#
#=================== ende von pathtable =======================
```

domaintable

Mit *domaintable* wird im allgemeinen ein bestimmtes Verhalten erzwungen, nachdem ein DNS-Lookup durchgeführt wurde. Diese Tabelle ermöglicht es dem Administrator, jeweils einen kurzen »Spitznamen« für häufig angesprochene Systeme oder Domains einzugeben, der automatisch in den eigentlichen Namen überführt wird. Sie können die Tabelle auch einsetzen, um falsche Host- oder Domainnamen durch die »richtigen« Informationen zu ersetzen.

Die meisten Sites benötigen keine *domaintable*-Einträge.

Das folgende Beispiel zeigt, wie eine inkorrekte Adresse, an die Leute Mail verschicken, durch die richtige Adresse ersetzt wird:

```
#============= /usr/local/lib/mail/domaintable =================
#
#
brokenhost.richtige.domain         brokenhost.falsche.domain
#
#
#=================== ende von domaintable =======================
```

aliases

Aliase bieten Ihnen eine ganze Reihe von Möglichkeiten:

- Sie bieten einen kurzen bzw. bekannten Namen für Mails an, die an eine oder mehrere Personen verschickt werden sollen.
- Sie führen ein Programm aus, wobei die Nachricht als Eingabe verwendet wird.
- Sie senden Nachrichten an eine Datei.

Alle Systeme benötigen Aliase für **Postmaster** und **MAILER-DAEMON**, damit sie die Regeln der RFC erfüllen.

Achten Sie immer besonders dann auf die Sicherheit, wenn Sie ein Alias definieren, das ein Programm ausführt oder an ein Programm schreibt, weil *sendmail* immer mit Setuid **root** läuft.

Änderungen in *aliases* treten erst in Kraft, nachdem der Befehl

```
# /usr/lib/sendmail -bi
```

ausgeführt wurde, der die benötigten *dbm*-Tabellen erzeugt. Dies kann auch mit dem Befehl *newaliases* erledigt werden, üblicherweise aus *cron* heraus.

Details zu Mail-Aliases können Sie der *aliases(5)*-Manpage entnehmen. Eine beispielhafte *aliases*-Datei ist in Beispiel 15--2 zu sehen.

Beispiel 15-2. *aliases-Beispieldatei*

#--------------------- /usr/local/lib/mail/aliases ------------------ # # demonstriert häufig verwendete Alias-Typen # usenet: janet # Alias für eine Person admin: joe,janet # Alias für mehrere Personen newspak-users: :include:/usr/lib/lists/newspak # lese Empfänger aus einer Datei changefeed: | /usr/local/lib/gup # Alias, das ein Programm ausführt complaints: /var/log/complaints # Alias, das Nachricht in Datei schreibt # # die beiden folgenden Aliases müssen vorhanden sein, um RFC-konform zu sein # es ist wichtig, daß bei der Auflösung eine Person herauskommt, die die Post # regelmäßig liest # postmaster: root # benötigter Eintrag MAILER-DAEMON: postmaster # benötigter Eintrag # #---

Selten benutzte Tabellen

Die folgenden Tabellen sind vorhanden, werden aber nur selten benutzt. Details können Sie der mit den Sendmail+IDA-Quellen ausgelieferten Dokumentation entnehmen.

uucprelays

Die Datei *uucprelays* wird verwendet, um den UUCP-Pfad zu besonders gut bekannten Sites abzukürzen. Auf diese Weise können Pfade mit mehreren Sprüngen bzw. unzuverlässige Pfade umgangen werden, die durch *pathalias* bei der Verarbeitung der UUCP-Maps erzeugt werden.

genericfrom und xaliases

Die Datei *genericfrom* versteckt lokale Benutzernamen und -Adressen vor der Außenwelt, indem lokale Benutzernamen automatisch in generische Absenderadressen umgewandelt werden, die keinem internen Benutzernamen entsprechen.

Das zugehörige *xalparse*-Utility automatisiert die Generierung der Dateien *genericfrom* und *aliases*, so daß die Übersetzung sowohl ein- als auch ausgehender Benutzernamen aus der Master-Datei *xaliases* erfolgt.

decnetxtable

Durch die Datei *decnetxtable* werden Adressen, die Domainnamen enthalten, in DECnet-Adressen umgewandelt. (Vergleichbar mit *domaintable*, die verwendet werden kann, um Adressen ohne Domainnamen in SMTP-Adressen umzuwandeln.)

sendmail installieren

Linux In diesem Abschnitt werfen wir einen Blick auf die Installation einer typischen Binary-Distribution von Sendmail+IDA und gehen die Schritte durch, die zur Lokalisierung und ordnungsgemäßen Funktion notwendig sind.

Die aktuelle Binär-Distribution von Sendmail+IDA für Linux ist auf **sunsite.unc.edu** unter */pub/Linux/system/Mail* zu finden. Selbst wenn Sie eine frühere Version von *sendmail* besitzen, möchte ich Ihnen wärmstens empfehlen, auf die Version *sendmail5.67b+IDA1.5* zu wechseln, weil alle Linux-spezifischen Patches bereits in den Sourcen vorhanden sind. Außerdem wurden verschiedene Sicherheitslöcher gestopft, die noch in den Versionen bis zum 1. Dezember 1993 vorhanden waren.

Wenn Sie *sendmail* aus den Sourcen erzeugen, sollten Sie den Anweisungen in den *README*s folgen, die in der Quell-Distribution enthalten sind. Der aktuelle Quellkode zu Sendmail+IDA ist auf **vixen.cso.uiuc.edu** zu finden. Um Sendmail+IDA unter Linux zum Laufen zu bringen, benötigen Sie auch die Linux-spezifischen Konfigurations-Dateien aus *newspak-2.2.tar.gz,* die auf **sunsite.unc.edu** im Verzeichnis*/pub/Linux/system/Mail* zu finden sind.

Wenn Sie bereits *smail* oder ein anderes Mailsystem installiert haben, sollten Sie die entsprechenden Dateien löschen (oder umbenennen), um ganz sicher zu gehen.

Auspacken der binären Distribution

Zuerst müssen Sie die Archiv-Datei an einer sicheren Stelle entpacken:

```
$ gunzip -c sendmail5.65b+IDA1.5+mailx5.3b.tgz | tar xvf -
```

Wenn Sie eine »moderne« *tar*-Version (z. B. aus einer neuen Slackware-Distribution) besitzen, können Sie auch einfach *tar -zxvf dateiname.tgz* eingeben und erzielen dasselbe Ergebnis.

Durch Entpacken des Archivs wird das Verzeichnis *sendmail5.65b+IDA1.5+ mailx5.3b* erzeugt. In diesem Verzeichnis finden Sie eine komplette Installation von Sendmail+IDA sowie eine Binary des *mailx*-Agenten. Alle Dateipfade unter diesem Verzeichnis spiegeln die Orte wieder, an denen die Dateien installiert sein sollten. Sie gehen also auf Nummer Sicher, wenn Sie die Dateien mit einem *tar*-Befehl übertragen:

```
# cd sendmail5.65b+IDA1.5+mailx5.3b
# tar cf -- . | (cd /; tar xvvpoof -)
```

Erzeugen von sendmail.cf

Um eine für Ihr System angepaßte *sendmail.cf*-Datei zu erzeugen, müssen Sie eine entsprechende *sendmail.m4*-Datei schreiben und mit *m4* verarbeiten. In */usr/local /lib/mail/CF* finden Sie eine Beispieldatei namens *sample.m4*. Kopieren Sie diese Datei und geben Sie ihr einen anderen Namen -- per Konvention wird *ihrhostname.m4* verwendet -- und editieren Sie sie so, daß die Situation auf Ihrer Site wiedergegeben wird.

Die Beispieldatei ist für eine reine UUCP-Site geschrieben, die Header mit Domainnamen verwendet und mit einem Smart Host kommuniziert. Bei solchen Sites müssen nur wenige Punkte bearbeitet werden.

In diesem Abschnitt möchte ich Ihnen eine kurze Übersicht der Makros geben, die Sie ändern müssen. Für eine komplette Beschreibung dessen, was sie tun, lesen Sie bitte die frühere Beschreibung von *sendmail.m4*.

`LOCAL_MAILER_DEF`

Definiert die Datei, die den Mailer für die lokale Nachrichtenauslieferung definiert. Was hier zu stehen hat, wird im Abschnitt »Definition des lokalen Mailers« beschrieben.

`PSEUDONYMS`

Definiert alle Namen, unter denen Ihr lokaler Host bekannt ist.

`DEFAULT_HOST`

An dieser Stelle muß Ihr voll qualifizierter Domainname stehen. Dieser Name erscheint als Ihr Hostname in allen ausgehenden Nachrichten.

`UUCPNAME`

An dieser Stelle steht Ihr unqualifizierter Hostname.

`RELAY_HOST` und `RELAY_MAILER`

Wenn Sie mit einem Smart Host über UUCP kommunizieren, muß `RELAY_HOST` auf den UUCP-Namen Ihres benachbarten »smarten Verteilerhost« gesetzt sein. Verwenden Sie den *UUCP-A*-Mailer, wenn Domainnamen im Header erscheinen sollen.

`DEFAULT_MAILER`

Wenn Sie im Internet hängen und über DNS kommunizieren, sollte dieser Wert auf `TCP-A` stehen. Dies weist

sendmail an, den *TCP-A*-Mailer zu verwenden, der Nachrichten über SMTP ausliefert und die RFC-konforme Adressierung verwendet. Bei Internet-Sites muß `RELAY_HOST` oder `RELAY_MAILER` üblicherweise nicht definiert sein.

Um die Datei *sendmail.cf* zu erzeugen, müssen Sie den folgenden Befehl ausführen:

```
# make ihrhostname.cf
```

Das verarbeitet Ihre *ihrhostname.m4*-Datei und erzeugt daraus *ihrhostname.cf*.

Als nächstes sollten Sie überprüfen, ob die von Ihnen erzeugte Konfigurations-Datei auch das tut, was Sie von ihr erwarten. Dies wird in den folgenden beiden Abschnitten behandelt.

Sind Sie mit den Ergebnissen zufrieden, müssen Sie sie mit dem folgenden Befehl an die richtige Stelle kopieren:

```
# cp ihrhostname.cf /etc/sendmail.cf
```

An diesem Punkt ist das *sendmail*-System bereit, in Aktion zu treten. Nehmen Sie die folgende Zeile in die entsprechende Startup-Datei (üblicherweise */etc/rc.inet2*) auf. Sie können sie auch von Hand ausführen, damit der Prozeß direkt gestartet wird:

```
# /usr/lib/sendmail -bd -q1h
```

Testen der Datei sendmail.cf

Um *sendmail* im Testmodus zu starten, müssen Sie es mit der Option *-bt* ausführen. Die Standard-Konfigurationsdatei ist die auf Ihrem System installierte Datei *sendmail.cf*. Eine andere Datei können Sie über die Option *-Cdateiname* angeben.

In den folgenden Beispielen testen wir *vstout.cf*, die Konfigurations-Datei, die aus der in Beispiel 15--1 abgebildeten *sendmail.m4* erzeugt wurde. .

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
>
```

Mit den folgenden Tests stellen wir sicher, daß *sendmail* in der Lage ist, Nachrichten an die Benutzer Ihres Systems auszuliefern. In allen Fällen sollte das Ergebnis des Tests gleich sein und auf das lokale System mit dem `LOCAL`-Mailer zeigen.

Zuerst wird getestet, wie Mail an einen lokalen Benutzer ausgeliefert wird:

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 me
rewrite: ruleset  3   input: me
rewrite: ruleset  7   input: me
rewrite: ruleset  9   input: me
rewrite: ruleset  9 returns: < me >
rewrite: ruleset  7 returns: < > , me
rewrite: ruleset  3 returns: < > , me
rewrite: ruleset  0   input: < > , me
rewrite: ruleset  8   input: < > , me
rewrite: ruleset 20   input: < > , me
```

```
rewrite: ruleset 20 returns: < > , @ vstout . vbrew . com , me
rewrite: ruleset  8 returns: < > , @ vstout . vbrew . com , me
rewrite: ruleset 26   input: < > , @ vstout . vbrew . com , me
rewrite: ruleset 26 returns: $# LOCAL $@ vstout . vbrew . com $: me
rewrite: ruleset  0 returns: $# LOCAL $@ vstout . vbrew . com $: me
```

Die Ausgabe zeigt, wie *sendmail* die Adresse intern verarbeitet. Diese wird von verschiedenen Regeln analysiert, an andere Regeln übergeben und in ihre einzelnen Komponenten aufgebrochen.

In unserem Beispiel wird die Adresse *me* an die Regelsätze 3 und 0 übergeben (das ist die Bedeutung der Eingabe *3,0* vor der Adresse). Die letzte Zeile zeigt die übergebene Adresse, wie sie vom Regelsatz 0 zurückgeliefert wird. Enthalten ist der Mailer, an den die Nachricht ausgeliefert würde sowie der an den Mailer übergebene Host- und Benutzername.

Als nächstes kommt die Prüfung von Nachrichten an einen Benutzer Ihres Systems mit UUCP-Syntax.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 vstout!me
rewrite: ruleset  3   input: vstout ! me
[...]
rewrite: ruleset  0 returns: $# LOCAL $@ vstout . vbrew . com  $: me
>
```

Als nächstes wird Mail an einen Benutzer Ihres Systems in Internet-Syntax mit Ihrem voll qualifizierten Host-Namen getestet:

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 me@vstout.vbrew.com
rewrite: ruleset  3   input: me @ vstout . vbrew . com
[...]
rewrite: ruleset  0 returns: $# LOCAL $@ vstout . vbrew . com $: me
>
```

Sie sollten die beiden vorherigen Tests mit jedem unter `PSEUDONYMS` und `DEFAULT_NAME` in ihrer *sendmail.m4*-Datei angegebenen Namen wiederholen:

Zum Schluß wird geprüft, ob Sie Nachrichten an Ihren Verteilerhost schicken können:

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 fred@moria.com
rewrite: ruleset  3   input: fred @ moria . com
rewrite: ruleset  7   input: fred @ moria . com
rewrite: ruleset  9   input: fred @ moria . com
rewrite: ruleset  9 returns: < fred > @ moria . com
rewrite: ruleset  7 returns: < @ moria . com > , fred
rewrite: ruleset  3 returns: < @ moria . com > , fred
rewrite: ruleset  0   input: < @ moria . com > , fred
rewrite: ruleset  8   input: < @ moria . com > , fred
rewrite: ruleset  8 returns: < @ moria . com > , fred
```

```
rewrite: ruleset 29   input: < @ moria . com > , fred
rewrite: ruleset 29 returns: < @ moria . com > , fred
rewrite: ruleset 26   input: < @ moria . com > , fred
rewrite: ruleset 25   input: < @ moria . com > , fred
rewrite: ruleset 25 returns: < @ moria . com > , fred
rewrite: ruleset  4   input: < @ moria . com > , fred
rewrite: ruleset  4 returns: fred @ moria . com
<?troff .Nd 6>
rewrite: ruleset 26 returns: < @ moria . com > , fred
rewrite: ruleset  0 returns: $# UUCP-A $@ moria $: < @ moria . com > , fred
>
```

Integrationstest zwischen sendmail.cf und den Tabellen

An diesem Punkt haben Sie sichergestellt, daß Nachrichten das gewünschte Standardverhalten zeigen, und daß Sie gültig adressierte Nachrichten senden und empfangen können. Um die Installation zu vollenden, könnte es notwendig sein, die entsprechenden *dbm*-Tabellen zu erzeugen, um die gewünschten Endergebnisse zu erzielen.

Nachdem Sie die für Ihre Site notwendige(n) Tabelle(n) angelegt haben, müssen Sie sie mit *dbm* durch Eingabe von *make* verarbeiten, wobei Sie sich in dem Verzeichnis befinden müssen, in dem die Tabellen gespeichert sind.

Wenn Sie nur mit UUCP arbeiten, müssen Sie *keine* der in *README.linux* angegebenen Tabellen erzeugen. Sie müssen die Dateien nur mit *touch* anstoßen, damit das Makefile funktioniert.

Wenn Sie nur mit UUCP arbeiten, aber neben dem Smart Host auch noch mit weiteren Sites kommunizieren, müssen Sie *uucpxtable*-Einträge für jede Site definieren (weil deren Mail sonst über den Smart Host ausgeliefert würde) und dann *dbm* über die aktualisierte *uucpxtable* laufen lassen.

Zuerst müssen Sie sicherstellen, daß Mail über Ihren `RELAY_HOST` über den `RELAY_MAILER` an diese Sites verschickt wird:

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 fred@sesame.com
rewrite: ruleset  3   input: fred @ sesame . com
rewrite: ruleset  7   input: fred @ sesame . com
rewrite: ruleset  9   input: fred @ sesame . com
rewrite: ruleset  9 returns: < fred > @ sesame . com
rewrite: ruleset  7 returns: < @ sesame . com > , fred
rewrite: ruleset  3 returns: < @ sesame . com > , fred
rewrite: ruleset  0   input: < @ sesame . com > , fred
rewrite: ruleset  8   input: < @ sesame . com > , fred
rewrite: ruleset  8 returns: < @ sesame . com > , fred
rewrite: ruleset 29   input: < @ sesame . com > , fred
rewrite: ruleset 29 returns: < @ sesame . com > , fred
rewrite: ruleset 26   input: < @ sesame . com > , fred
rewrite: ruleset 25   input: < @ sesame . com > , fred
rewrite: ruleset 25 returns: < @ sesame . com > , fred
rewrite: ruleset  4   input: < @ sesame . com > , fred
rewrite: ruleset  4 returns: fred @ sesame . com
rewrite: ruleset 26 returns: < @ sesame . com > , fred
rewrite: ruleset  0 returns: $# UUCP-A $@ moria $: < @ sesame . com > , fred
>
```

Wenn Sie neben `RELAY_HOST` weitere UUCP-Nachbarn haben, müssen Sie sicherstellen, daß Mail an diese sich auch entsprechend verhält. Mail-Adressen in UUCP-Syntax, die an einen Host gerichtet sind, mit dem Sie über UUCP kommunizieren, sollten direkt an diesen weitergeleitet werden (solange Sie das nicht durch einen expliziten *domaintable*-Eintrag verhindern). Nehmen wir an, daß der Host swim einer Ihrer direkten UUCP-Nachbarn ist. Die Angabe von *swim!fred* an *sendmail* sollte dann das folgende Ergebnis erzeugen:

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 swim!fred
rewrite: ruleset  3   input: swim ! fred
[...lines omitted...]
rewrite: ruleset  0 returns: $# UUCP $@ swim $: < > , fred
>
```

Wenn Sie *uucpxtable*-Einträge besitzen, mit denen die Auslieferung an bestimmte UUCP-Nachbarn erzwungen wird, die ihre Mail mit Internet-eigenen Headern versenden, in denen Domainnamen enthalten sind, sollten Sie dies auch prüfen:

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 dude@swim.2birds.com
rewrite: ruleset  3   input: dude @ swim . 2birds . com
[...lines omitted...]
rewrite: ruleset  0 returns: $# UUCP $@ swim . 2birds $: < > , dude
>
```

Administrivialitäten und dumme Mail-Tricks

Nachdem wir nun die Konfiguration, Installation und den Test eines Sendmail+IDA-Systems theoretisch besprochen haben, sollten wir auch einen Blick darauf werfen, was sich so im normalen Leben eines Mail-Administrators wirklich ereignet.

Systeme stürzen manchmal ab. Modems oder Telefonleitungen funktionieren manchmal nicht. DNS-Definitionen sind manchmal falsch eingestellt. Netzwerke hängen sich unerwartet auf. In solchen Fällen müssen Mail-Administratoren wissen, wie man schnell, effektiv und *sicher* reagiert, damit der Nachrichtenfluß über alternative Routen weiterläuft, bis das entfernte System oder der Service-Provider den normalen Betrieb wieder aufnimmt.

Der Rest dieses Kapitels soll Sie mit Lösungen versorgen, die häufig bei Mail-Notfällen angewendet werden.

Weiterleitung von Nachrichten an einen Verteilerhost

Um die Mail für einen bestimmten Host oder für eine bestimmte Domain an ein dafür vorgesehenes Verteilersystem weiterzuleiten, wird im allgemeinen die Datei *mailertable* verwendet. Um beispielsweise die Mail für **backwood.org** zu deren UUCP-Gateway **backdoor** weiterzuleiten, müssen Sie den folgenden Eintrag in *mailertable* aufnehmen:

```
 UUCP-A,backdoor   backwood.org
```

Auslieferung von Mail an fehlerhaft konfigurierte Sites erzwingen

Internet-Hosts haben häufig Probleme, Mail an fehlerhaft konfigurierte Sites zu übergeben. Es gibt verschiedene Varianten dieses Problems, aber das allgemeine Symptom ist, daß Mail vom anderen System zurückgeschickt (gebounced) wird oder aber überhaupt nicht dort ankommt.

Diese Probleme können den lokalen Systemadministrator in eine schlechte Position bringen, weil sich die Benutzer im allgemeinen nicht darum kümmern, daß er nicht jedes System auf der ganzen Welt verwaltet (oder weiß, wie man den Administrator des anderen Systems bittet, das Problem zu beheben). Er weiß nur, daß die Nachrichten den gewünschten Empfänger nicht erreichen und daß Sie die greifbarste Person sind, bei der er sich beschweren kann.

Die Konfiguration der anderen Site ist deren Problem, nicht Ihres. Auf keinen Fall sollten Sie Ihre Konfiguration demolieren, um mit der fehlerhaft konfigurierten Site kommunizieren zu können. Falls Sie den Postmaster des anderen Systems nicht dazu bewegen können, die Konfiguration in einer angemessenen Zeit zu korrigieren, bleiben Ihnen zwei Möglichkeiten:

- Es ist generell möglich, Mail erfolgreich an das andere System auszuliefern. Weil das andere System aber falsch konfiguriert ist, gibt es möglicherweise keine Antworten auf die Nachrichten, aber das ist dann endgültig das Problem des anderen Administrators.

  Sie können die fehlerhaften Header in Ihren ausgehenden Nachrichten nur korrigieren, indem Sie einen entsprechenden *domaintable*-Eintrag für deren Host /Domain benutzen, der dafür sorgt, daß die ungültigen Informationen bei von Ihrer Seite ausgehenden Nachrichten korrigiert werden:

  ```
  braindead.correct.domain.com        braindead.wrong.domain.com
  ```

- Häufig schickt eine fehlerhaft konfigurierte Site Nachrichten an das sendende System zurück, wobei als Fehler sinngemäß »diese Nachricht ist nicht für dieses System« zurückgeliefert wird, weil in der Konfiguration die `PSEUDONYMNS` oder ähnliches nicht korrekt eingestellt wurden. Es ist möglich, jegliche Host- und Domain-Information aus den Nachrichten zu entfernen, die von Ihrer Site an die andere geschickt werden.

  Das `!` im folgenden *mailertable* liefert Nachrichten an die andere Site aus und läßt sie deren *sendmail* gegenüber so erscheinen, als käme sie direkt von ihrem eigenen System. Das ändert übrigens nur die Envelope-Adresse, die korrekte Absenderadresse bleibt in der Nachricht erhalten.

  ```
  TCP!braindead.richtig.domain.com   braindead.falsch.domain.com
  ```

Selbst wenn es Ihnen nun gelingt, die Nachrichten an deren System auszuliefern, wird man dort nicht unbedingt in der Lage sein, darauf zu antworten (schließlich ist das System ja fehlerhaft konfiguriert). Andererseits werden dann die Benutzer des anderen Systems ihre Administratoren anmachen, nicht Ihre Benutzer Sie.

Auslieferung von Mail über UUCP erzwingen

In einer (aus der Sicht des Internet) idealen Welt besitzen alle Hosts Einträge im Domain Name Service und senden Nachrichten mit voll qualifizierten Domainnamen.

Soll über UUCP mit einer solchen Site kommuniziert werden, können Sie dafür sorgen, daß Nachrichten über die direkte UUCP-Verbindung und nicht über den standardmäßigen Mailer ausgeliefert werden, indem Sie mit Hilfe der *uucpxtable* die Domainnamen aus der Nachricht entfernen.

Um die Auslieferung an **sesame.com** über UUCP zu erledigen, würden Sie die folgende Zeile in Ihre *uucpxtable* aufnehmen:

```
#Domain aus sesame.com entfernen, um Auslieferung über UUCP zu erzwingen
sesame    sesame.com
```

Das Ergebnis ist, daß sendmail dann erkennt (über `UUCPNODES` in der Datei *sendmail.m4*), daß Sie direkt mit dem anderen System verbunden sind, woraufhin die Nachrichten in die Queue geschoben werden, um mit UUCP ausgeliefert zu werden.

Auslieferung von Nachrichten über UUCP verhindern

Aber auch der entgegengesetzte Fall kommt vor. Häufig besitzen Systeme eine Reihe von direkten UUCP-Verbindungen, die nur selten benutzt werden, nicht so zuverlässig sind, oder nicht immer als standardmäßiger Mailer- oder Verteilerhost zur Verfügung stehen.

Beispielsweise gibt es im Raum Seattle eine ganze Reihe von Systemen, die die verschiedenen Linux-Distributionen über »anonymous UUCP« austauschen, sobald diese Distributionen erscheinen. Diese Systeme unterhalten sich über UUCP nur, wenn nötig. Daher ist es grundsätzlich schneller und sicherer, Nachrichten über mehrere, sehr sichere Sprünge und gängige (und immer verfügbare) Verteilerhosts zu senden.

Die Auslieferung von Nachrichten über UUCP an einen Host zu unterdrücken ist sehr einfach, wenn Sie über eine direkte Verbindung zu diesem Host verfügen. Besitzt das andere System einen voll qualifizierten Domainnamen, können Sie einen Eintrag wie den folgenden in die *domaintable* aufnehmen:

```
# Auslieferung von Mail über UUCP an einen Nachbarn unterdrücken
snorkel.com       snorkel
```

Das ersetzt jedes Auftreten des UUCP-Namens durch den voll qualifizierten Domainnamen und verhindert, daß die `UUCPNODES`-Zeile in der *sendmail.m4*-Datei greift. Das Ergebnis ist, daß Nachrichten nun über `RELAY_MAILER` und `RELAY_HOST` (oder `DEFAULT_MAILER`) ausgeliefert werden.

sendmail-Queue auf Anforderung ausführen

Um Nachrichten direkt zu verarbeiten, die sich in der Queue angesammelt haben, brauchen Sie nur */usr/lib/runq* einzugeben. Auf diese Weise wird *sendmail* mit den entsprechenden Optionen gestartet, die benötigt werden, um die Queue mit den offenen Jobs direkt zu bearbeiten, anstatt den nächsten vorgesehenen Durchlauf abzuwarten.

Ausgabe von Statistiken

Viele Site-Administratoren (und die Personen, für die sie arbeiten) sind an dem Volumen von Mails interessiert, die von, zur und über die lokale Site fließen. Es gibt eine Anzahl von Möglichkeiten, Mail-Transfer zu quantifizieren:

- *sendmail* wird mit einem Utility namens *mailstats* ausgeliefert, das eine Datei namens */usr/local/lib/mail/sendmail.st* liest und einen Bericht der übertragenen Nachrichten und Bytes von jedem in der *sendmail.cf* eingetragenen Mailer erstellt. Diese Datei muß vom lokalen Administrator von Hand angelegt werden, damit *sendmail* auch ein entsprechendes Logging durchführt. Die fortlaufenden Gesamtzahlen werden gelöscht, indem die Datei *sendmail.st* gelöscht und wieder neu angelegt wird. Nachfolgend wird eine Möglichkeit aufgezeigt:

  ```
  # cp /dev/null /usr/lib/local/mail/sendmail.st
  ```

- Der wohl beste Weg, qualitative Berichte darüber zu erzeugen, wer Mail verwendet, und wieviel Volumen an und über das lokale System sowie von ihm ausgehend transferiert wird, besteht darin, das Mail-Debugging mit *syslogd* zu aktivieren. Das bedeutet allgemein, daß der */etc/syslogd*-Dämon von Ihrem Startup-Script aus gestartet werden muß (was Sie sowieso tun sollten). In der Datei */etc/syslog.conf* müssen Sie eine Zeile einfügen, die etwa wie folgt aussieht:

  ```
  mail.debug                         /var/log/syslog.mail
  ```

  Wenn Sie `mail.debug` verwenden und ein mittleres bis hohes Mail-Aufkommen haben, kann die syslog-Ausgabe recht groß sein. Die *syslogd*-Ausgabedateien sollten regelmäßig aus *crond* ausgetauscht oder gesäubert werden.

  Es gibt eine ganze Anzahl gängiger Utilities, mit denen die Ausgabe des Mail-Loggings *syslogd* summiert werden kann. Eines der bekannteren Utilities ist *syslog-stat.pl*, ein *perl*-Script, das mit den Sendmail+IDA-Quellen verteilt wird.

Mischen und Abstimmen binärer Distributionen



Es gibt keine Standard-Konfiguration für den Transport elektronischer Post und der entsprechenden Programme,

und es gibt keine »einzig wahre Verzeichnisstruktur«.

Dementsprechend ist es wichtig, sicherzustellen, daß die verschiedenen Teile des Systems (Usenet-News, Mail, TCP/IP) mit der Position des lokalen Mail-Auslieferungsprogramms (*lmail*, *deliver* etc.), mit dem entfernten Mail-Auslieferungsprogramm (*rmail*) und mit dem Mail-Transportprogramm (*sendmail* oder *smail*) zusammenpassen. Solche Annahmen sind grundsätzlich nicht dokumentiert, obwohl Ihnen der Befehl *strings* helfen kann, zu ermitteln, welche Dateien und Verzeichnisse erwartet werden. Nachfolgend einige Probleme, denen wir in der Vergangenheit bei einigen der gängigen Linux-Binärdistributionen und Sourcen begegnet sind.

- Manche Versionen der NET-2-Distribution von TCP/IP definieren die Dienste für ein Programm namens *umail* und nicht für *sendmail*.
- Verschiedene Ports von *elm* und *mailx* verwenden den Auslieferungsagenten */usr/bin/smail* und nicht *sendmail*.
- Sendmail+IDA hat einen lokalen Mailer namens *deliver* bereits integriert, erwartet ihn aber in */bin* und nicht in der für Linux eigentlich typischeren Position */usr/bin*.

Statt uns nun mit den Problemen herumzuschlagen, die die Kompilierung aller Mail-Clients mit sich bringen würde, umgehen wir dies durch Einrichten entsprechender Links.

Wo Sie weitere Informationen finden

Weitere Informationen zu *sendmail* finden Sie im »Linux Electronic Mail HOWTO«, das regelmäßig in *comp.answers* gepostet wird. Es steht auch über FTP auf **rtfm.mit.edu** zur Verfügung.

Der definitive Ort für Informationen sind aber die Sendmail+IDA-Quellen. Sehen Sie sich im Verzeichnis *ida/cf*, unter dem Quellverzeichnis, die Dateien *DBM-GUIDE*, *OPTIONS* und *Sendmail.mc* an. Lesen Sie mehr dazu im Kapitel 15.

Fußnoten

(1)

deliver wurde von Chip Salzenberg (chip%tct@ateng.com) geschrieben. Es ist Teil verschiedener Linux-Distributionen und kann über die üblichen FTP-Archive wie ftp.uu.net bezogen werden.

Wünschen Sie mehr Informationen zu der gedruckten Version des Buches "Linux: Wegweiser durch das Netzwerk" dann klicken Sie hier.

Kapitel 16
Netnews

Geschichte des Usenet

Die Idee der Netzwerk-News wurde im Jahr 1979 geboren. Die beiden Studenten Tom Truscott und Jim Ellis wollten UUCP verwenden, um Maschinen untereinander zu verbinden, damit Informationen zwischen UNIX-Benutzern ausgetauscht werden könnten. Sie richteten ein kleines Netzwerk aus drei Maschinen in North Carolina ein.

Zu Beginn wurde die Datenübertragung durch eine Reihe von Shell-Scripten (die später in C umgeschrieben wurden) gehandhabt. Diese wurden der Öffentlichkeit aber niemals zugänglich gemacht, sondern schnell durch »A«-News ersetzt, der ersten öffentlichen Release einer News-Software.

A News war nicht dafür ausgelegt, mehr als einige Artikel pro Gruppe und Tag zu verwalten. Als das Volumen wuchs, wurde es von Mark Horton und Matt Glickman neu geschrieben, die es die »B«-Release (also B News) tauften. Die erste öffentliche Release von B News war Version 2.1 im Jahr 1982. Es wurde kontinuierlich erweitert, und neue Features wurden integriert. Die aktuelle Version ist B News 2.11. Es ist aber langsam überholt; der letzte offizielle Verwalter wechselte zu INN.

Das System wurde noch einmal neu geschrieben und im Jahr 1987 von Geoff Collyer und Henry Spencer als Release »C,« oder C News veröffentlicht. Seit seiner Veröffentlichung hat es eine Reihe von Patches für C News gegeben. Der bekannteste ist sicherlich die »C News Performance Release«. Bei Sites mit einer großen Anzahl von Gruppen ist der durch das ständige Aufrufen von *relaynews* erzeugte Overhead beträchtlich. Dieses Programm ist für die Weiterleitung eingehender Artikel an andere Hosts zuständig. Die Performance Release erweitert *relaynews* um eine Option, mit der das System im *Daemon-Modus* ausgeführt werden kann, bei dem sich das Programm selbständig in den Hintergrund begibt. Die Performance Release ist die Version der C News, die momentan in den meisten Linux-Releases zu finden ist. Seit Herbst 1995 gibt es das sogenannte Cleanup Release, das aber keine neuen Features einführt, sondern nur einige Teile verbessert.

Alle News-Releases bis hin zu C waren primär für die Verwendung in UUCP-Netzwerken ausgelegt, obwohl sie auch in anderen Umgebungen verwendet werden konnten. Die effiziente Übertragung von

News über Netzwerke wie TCP/IP oder DECNet machte ein neues Schema notwendig. Daher wurde im Jahr 1986 das *Network News Transfer Protocol* (NNTP) eingeführt. Es basiert auf Netzwerkverbindungen und spezifiziert eine Reihe von Befehlen für die interaktive Übertragung und den Empfang von Artikeln.

Über das Netz stehen eine ganze Reihe von NNTP-basierten Anwendungen zur Verfügung. Eine davon ist das *nntpd*-Paket von Brian Barber und Phil Lapsley, mit dem Sie Newsreading-Dienste für eine ganze Reihe von Hosts innerhalb eines lokalen Netzwerks bereitstellen können. *nntpd* ist so entworfen worden, daß es News-Pakete wie B News oder C News um NNTP-Features erweitert.

Ein anderes NNTP-Paket ist INN, oder auch *Internet News*. Es ist nicht einfach nur ein Frontend, sondern ein vollständiges News-System. Es besitzt einen leistungsfähigen News-Verteilerdämon, der effektiv mehrere NNTP-Links gleichzeitig verwalten kann und ist daher *der* News-Server auf vielen Internet-Sites.

Und was ist nun das Usenet?

Eine der erstaunlichsten Eigenschaften des Usenet ist, daß es weder Teil einer Organisation ist, noch irgendeine Form zentraler Autorität besitzt. Tatsächlich können Sie außer der technischen Beschreibung nicht definieren, *was* es ist; Sie können nur sagen, was es nicht ist.

Auch auf die Gefahr hin, daß sich das dumm anhört, können Sie das Usenet als einen Zusammenschluß separater Sites betrachten, die untereinander Usenet-News austauschen. Um selbst zu einer Usenet-Site zu werden, müssen Sie nur eine andere Usenet-Site finden und mit deren Besitzern und Verwaltern eine Vereinbarung darüber treffen, daß mit Ihnen News ausgetauscht werden. Das Weitergeben von News an eine andere Site wird auch als *Feeding* bezeichnet.

Die grundlegende Einheit der Usenet-News ist der Artikel. Dabei handelt es sich um eine Nachricht, die von einem Benutzer geschrieben und ins Netz eingespeist (»gepostet«) wurde. Damit das News-System in der Lage ist, mit Artikeln umzugehen, werden diesen administrative Informationen, die sogenannten Artikel-Header, vorangestellt. Ein solcher Header erinnert stark an das Mail-Header-Format, das im Internet-Mail-Standard-RFC 822 beschrieben ist. Er besteht aus verschiedenen Textzeilen, die mit einem durch einen Doppelpunkt abgeschlossenen Feldnamen beginnen, dem dann der Wert des Feldes folgt.(1)

Artikel werden in eine oder mehrere *Newsgruppen* übertragen. Sie können sich Newsgruppen als ein Forum für Artikel zu einem bestimmten Thema vorstellen. Alle Newsgruppen sind in einer Hierarchie angeordnet, wobei der Gruppenname den Platz in der Hierarchie widerspiegelt. Das macht es häufig einfach zu erkennen, um was es in einer Gruppe eigentlich geht. Beispielsweise kann jeder aus dem Namen dieser Newsgruppe erkennen, daß *comp.os.linux.announce* für Ankündigungen zu einem Computer-Betriebssystem namens Linux genutzt wird.

Diese Artikel werden dann zwischen allen Usenet-Sites ausgetauscht, die willens sind, News aus dieser Gruppe weiterzuleiten. Sind zwei Sites übereingekommen, News miteinander auszutauschen, können sie jede gewünschte Newsgruppe transportieren. Es können sogar eigene lokale News-Hierarchien eingeführt werden. Beispielsweise könnte **groucho.edu** einen News-Link zu **barnyard.edu** besitzen, der einer der Haupt-Newsverteiler ist, sowie verschiedene Links zu kleineren Sites, die es mit News versorgt. Nun könnte das Barnyard College alle Usenet-Gruppen empfangen wollen, während die GMU nur einige

der Haupthierarchien wie *sci*, *comp*, *rec* etc. nutzen möchte. Einige dahinterliegende Sites, z. B. eine UUCP-Site namens **brewhq**, könnten noch weniger Gruppen wollen, weil die Netzwerk- oder Hardware-Ressourcen nicht ausreichen. Andererseits könnte **brewhq** Newsgruppen aus der *fj*-Hierarchie empfangen wollen, die von der GMU nicht empfangen werden. **brewhq** besitzt daher einen weiteren Link zu **gargleblaster.com**, die alle *fj*-Gruppen empfängt und diese an **brewhq** weitergibt. Der Datenfluß der News ist in Abbildung 16--1 zu sehen.

Abbildung 16-1. Datenfluß von Usenet-News durch die Groucho-Marx-Universität

Die Bezeichnungen neben den von **brewhq** ausgehenden Pfeilen bedürfen möglicherweise der Erklärung. Standardmäßig sollen alle lokalen generierten News an **groucho.edu** geschickt werden. Weil **groucho.edu** aber die *fj*-Gruppen nicht transportiert, macht es keinen Sinn, ihm Nachrichten dieser Gruppe zu schicken. Daher wird der Feed von **brewhq** zur GMU mit dem Text *all,!fj* bezeichnet, was verdeutlichen soll, daß alle Gruppen außer denen unterhalb von *fj* dorthin gesendet werden.

Wie behandelt Usenet die News?

Heutzutage hat das Usenet enorme Proportionen angenommen. Sites, die die gesamten Netnews vorhalten, transportieren täglich weit über 60 Megabyte.(2) Natürlich geht es hier um weit mehr, als um das reine Umherschieben von Dateien. Sehen wir uns also an, wie die meisten UNIX-Systeme Usenet-News behandeln.

News werden im Netz über verschiedene Wege transportiert. Das historische Medium war UUCP, aber heutzutage wird der Hauptdatenverkehr über Internet-Sites abgewickelt. Das zum Routen verwendete Verfahren wird als *Flooding-Algorithmus* bezeichnet: Jede Site verwaltet eine Anzahl von Links (*News-Feeds*) zu anderen Sites. Jeder vom lokalen System generierte oder empfangene Artikel wird an diese Links weitergeleitet. Wurde er bereits ausgeliefert, wird er ausrangiert. Eine Site kann sich ansehen, welche Sites der Artikel bereits passiert hat, indem sie das Header-Feld `Path:` untersucht. Dieser Header enthält eine Liste aller Systeme, die den Artikel weitergeleitet haben, in Bang-Path-Notation.

Um Artikel unterscheiden und Duplikate erkennen zu können, enthalten Usenet-Artikel eine sog. Message-ID (spezifiziert im Header-Feld `Message-Id:`). Diese kombiniert den Namen der Site, auf welcher der Artikel gepostet wurde, und eine Seriennummer in einer Zeichenkette des Formats <*seriennummer@site*>. Bei jedem verarbeiteten Artikel speichert das News-System diesen ID in einer *history*-Datei, über die alle neu eingegangenen Artikel abgeprüft werden.

Der Fluß zwischen diesen beiden Sites kann über zwei Kriterien eingeschränkt werden. Zum einen kann einem Artikel eine bestimmte »Distribution« (im Header-Feld `Distribution:`) zugewiesen werden, die verwendet werden kann, um ihn auf eine bestimmte Gruppe von Sites zu beschränken. Auf der anderen Seite kann die Anzahl der ausgetauschten Newsgruppen durch das sendende, aber auch durch das empfangende System eingeschränkt werden. Der Satz von Newsgruppen und Distributionen, die an eine bestimmte Site übertragen werden dürfen, ist üblicherweise in der Datei *sys* definiert.

Alleine die Anzahl der Artikel macht es üblicherweise notwendig, das obige Schema zu verbessern. Bei UUCP-Netzwerken ist es durchaus natürlich, Artikel über eine gewisse Zeit zu sammeln und in einer einzelnen Datei zu speichern, die komprimiert und dann an die andere Site übertragen wird. Die Technik

wird Stapelverarbeitung oder auch *Batching* genannt.

Eine alternative Technik ist das *ihave/sendme*-Protokoll, das verhindert, daß Artikel doppelt übertragen werden, und auf diese Weise Bandbreite spart. Anstatt alle Artikel in einer Batch-Datei zu speichern und zu versenden, werden nur die Message-IDs der Artikel zu einer großen »ihave«-Nachricht zusammengefasst und an die andere Site übertragen. Die andere Site liest diese Nachricht und vergleicht sie mit der eigenen History-Datei. Daraufhin wird in einer »sendme«-Nachricht eine Liste der gewünschten Artikel zurückgeliefert. Nur diese Artikel werden dann übertragen.

Diese Technik macht natürlich nur Sinn, wenn zwei große Sites Daten austauschen, die News jeweils über mehrere unabhängige Quellen beziehen und sich oft genug pollen, so daß ein effizienter Fluß von News möglich ist.

Auf Internet-Sites wird in der Regel TCP-IP-basierte Software eingesetzt, die das Network News Transfer Protocol (NNTP) verwendet. Diese überträgt News zwischen verschiedenen Quellen und bietet Usenet-Zugriff für einzelne Benutzer auf anderen Hosts.

NNTP kennt drei verschiedene Arten der News-Übertragung. Die erste ist eine Echtzeit-Version von ihave/sendme, die als »rüberschieben« (*pushing*) von News bezeichnet wird. Die zweite Technik ist das »rüberziehen« (*pulling*) von News. Dabei fordert der Client eine Liste von Artikeln einer bestimmten Newsgruppe oder -Hierarchie an, die bis zu einem bestimmten Zeitpunkt auf der Serverseite angekommen sind, und wählt dann die aus, die in der History-Datei nicht gefunden werden konnten. Bei der dritten Methode handelt es sich um interaktives Lesen von News. Dabei kann Ihr Newsreader Artikel aus angegebenen Newsgruppen empfangen und gleichzeitig Artikel mit unvollständigen Header-Informationen posten.

Bei jeder Site werden News in einer Verzeichnis-Hierarchie unter */var/spool/news* gehalten. Jeder Artikel steht in einer separaten Datei und jede Newsgruppe in einem separaten Verzeichnis. Der Verzeichnisname besteht aus dem Namen der Newsgruppe. Deren Komponenten bilden dabei die Komponenten des Pfads. Die Artikel der Newsgruppe *comp.os.linux.misc* befinden sich also im Verzeichnis */var /spool/news/comp/os/linux/misc*. Den Artikeln einer Newsgruppe werden Zahlen in der Reihenfolge ihres Eintreffens zugewiesen. Diese Zahlen dienen als Dateinamen. Der Wertebereich von Artikeln, die gerade online sind, wird in einer Datei namens *active* gespeichert. Die Datei enthält außerdem eine Liste der Ihrer Site bekannten Newsgruppen.

Weil Festplattenplatz nur eine endliche Ressource darstellt,(3) muß man nach einer gewissen Zeit anfangen, alte Artikel auszusortieren. Das wird als *Expiring*, also »Löschen«, bezeichnet. Normalerweise werden Artikel aus bestimmten Gruppen und Hierarchien nach einer festen Zeit (in Tagen) für ungültig erklärt (und gelöscht). Diese Zeit kann durch den Verfasser durch Angabe eines Ungültigkeitsdatums im Feld `Expires:` des Artikel-Headers überschrieben werden.

Fußnoten

(1)

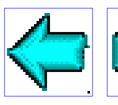
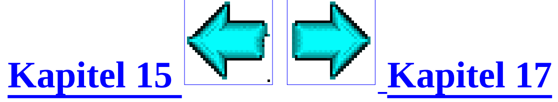
Das Format von Usenet-News-Nachrichten ist in RFC 1036, »Standard for interchange of USENET messages« spezifiziert.

(2)

Moment mal: 60 MByte bei 9600 Bps sind 60 Millionen durch 1200 ..., sind … murmel, murmel… Hey! Das sind 34 Stunden!

(3)

Manche Leute behaupten, das Usenet sei eine Verschwörung von Modem- und Festplattenherstellern.

Inhaltsverzeichnis  **Kapitel 15**  **Kapitel 17**

Kapitel 17
C News

Eines der bekanntesten Softwarepakete für Netnews ist C News. Es wurde für Sites entwickelt, die News über UUCP-Links verteilen. Dieses Kapitel behandelt die zentralen Konzepte von C News und bespricht die grundlegenden Installations- und Verwaltungsaufgaben.

C News speichert die meisten seiner Konfigurations-Dateien im Verzeichnis */usr /lib/news* und die meisten seiner Binaries im Verzeichnis */usr/lib/news/bin*. Artikel werden unter */var/spool/news* gehalten. Sie sollten sicherstellen, daß praktisch alle Dateien in diesem Verzeichnis dem Benutzer **news** und der Gruppe **news** angehören. Die meisten Probleme entstehen durch Dateien, auf die C News nicht zugreifen kann. Machen Sie es sich zur Regel, über den Befehl *su* der Benutzer **news** zu werden, bevor Sie irgendetwas in diesem Verzeichnis anfassen. Die einzige Ausnahme ist *setnewsids*, mit dem einige News-Programme Ihre echte Benutzer-ID einstellen. Dieses Programm muß **root** gehören und das Setuid-Bit gesetzt haben.

In diesem Kapitel beschreiben wir ausführlich die Konfigurations-Dateien von C News und zeigen Ihnen, was Sie tun müssen, um Ihre Site am Laufen zu halten.

News ausliefern

Artikel können an C News auf verschiedene Arten übergeben werden. Postet ein lokaler Benutzer einen Artikel, übergibt der Newsreader ihn üblicherweise an den Befehl *inews*, der die Header-Informationen dann vervollständigt. News von anderen Sites, gleichgültig, ob ein einziger Artikel oder ein ganzer Stapel (Batch), werden an den Befehl *rnews* übergeben, der sie im Verzeichnis */var/spool/news/in. coming* ablegt. Von dort werden sie zu einem späteren Zeitpunkt von *newsrun* übernommen. Bei beiden Techniken landet der Artikel aber schließlich beim Befehl *relaynews*.

Bei jedem Artikel überprüft der *relaynews*-Befehl zuerst, ob der Artikel bereits auf der lokalen Site gesehen wurde, indem er die Message-ID in der Datei *history* nachschaut. Doppelte Artikel werden aussortiert. Danach sieht sich *relaynews* die Header-Zeile `Newsgroups:` an, um herauszufinden, ob die lokale Site Artikel aus einer dieser Gruppen angefordert hat. Wenn ja, und wenn die News-Gruppe in der Datei *active* aufgeführt ist, versucht *relaynews* den Artikel im entsprechenden Verzeichnis des News-Spoolbereichs zu speichern. Wenn das Verzeichnis nicht existiert, wird es angelegt. Der Message-ID des Artikels wird dann in die *history*-Datei aufgenommen. Anderenfalls sortiert *relaynews* den Artikel aus.

Ist *relaynews* nicht in der Lage, einen eingehenden Artikel zu speichern, weil dieser in eine Gruppe gepostet wurde, die nicht in Ihrer *active*-Datei aufgeführt ist, wird der Artikel in der Gruppe *junk* untergebracht.(1) *relaynews* prüft auch, ob die Artikel veraltet sind bzw. ob ein fehlerhaftes Datum vorliegt, und sortiert sie

entsprechend aus. Eingehende Batches, die aus irgendeinem anderen Grund nicht bearbeitet werden können, werden nach */var/spool/news/in.coming/bad* verschoben, und eine entsprechende Fehlermeldung wird in die Log-Datei geschrieben.

Danach werden die Artikel an alle anderen Sites verteilt, die News aus diesen Gruppen anfordern. Dabei wird das für jede Site spezifizierte Transportverfahren benutzt. Um sicherzustellen, daß die Artikel nicht an eine Site übertragen werden, die diese bereits gesehen hat, wird jede Ziel-Site mit dem Header-Feld `Path:` des Artikels verglichen. Dieses enthält eine Liste der Sites in Bang-Pfad-Notation, die der Artikel bislang passiert hat. Nur wenn der Name der Ziel-Site nicht in dieser Liste erscheint, wird der Artikel dorthin gesendet.

C News wird üblicherweise verwendet, um News zwischen UUCP-Sites zu verteilen, obwohl es auch möglich ist, es in einer NNTP-Umgebung einzusetzen. Um News an eine andere Site auszuliefern -- gleichgültig, ob es sich dabei um einzelne Artikel oder komplette Batches handelt -- wird *uux* verwendet, das *rnews* auf dem anderen Rechner ausführt, wobei die Artikel oder der Batch über die Standardeingabe eingespeist werden.

Ist der Batch-Betrieb für eine bestimmte Site aktiviert, überträgt C News die eingehenden Artikel nicht direkt, sondern hängt den Pfadnamen an eine Datei (üblicherweise *out.going/site/togo*) an. Ein Batch-Programm wird regelmäßig über die *crontab* gestartet(2), das die Artikel in einer oder mehreren Dateien unterbringt, diese optional komprimiert und sie dann an *rnews* auf der anderen Site übergibt.

Abbildung 17--1 zeigt den Fluß von News durch *relaynews*. Artikel können an die lokale Site (bezeichnet durch `ME`), über E-Mail zu einer Site namens **ponderosa** und zu einer Site namens **moria** übertragen werden, für die der Batch-Betrieb aktiviert ist.

Abbildung 17-1. *Flußvon News durch relaynews*

Installation

Zur Installation von C News entpacken Sie zunächst das tar-Archiv mit allen Programmen und Konfigurationsdateien, wenn Sie dies nicht bereits getan haben. Dann müssen Sie die nachfolgend aufgeführten Konfigurations-Dateien editieren, die alle in */usr/lib/news* zu finden sind. Die einzelnen Dateiformate werden in den folgenden Abschnitten behandelt.

sys

Sie müssen wahrscheinlich die Zeile `ME` modifizieren, die Ihr System beschreibt, obwohl die Eingabe `all/all` immer eine sichere Sache ist. Sie müssen auch eine Zeile für jede Site hinzufügen, die Sie mit News versorgen.

Wenn Sie keine anderen Sites mit News versorgen, benötigen Sie nur eine Zeile, die alle lokal generierten Artikel an die Site übergibt, die Sie mit News versorgt (an Ihren sog. Feed). Nehmen wir an, daß **moria** Ihr Feed ist, dann sollte Ihre *sys*-Datei wie folgt aussehen:

```
ME:all/all::
moria/moria.orcnet.org:all/all,!local:f:
```

organization

Der Name Ihrer Organisation, beispielsweise »Virtuelle Brauerei GmbH«. Auf Ihrem Rechner zu Hause sollten Sie »private Site,« oder etwas ähnliches angeben. Für die meisten Leute ist Ihr Rechner nicht richtig konfiguriert, solange diese Datei nicht angepaßt wurde.

newsgroups

Eine Liste mit allen News-Gruppen sowie einer einzeiligen Beschreibung der jeweiligen Funktion. Die Beschreibungen werden häufig von Ihrem Newsreader verwendet, wenn eine Liste aller von Ihnen abonnierten Gruppen ausgegeben wird.

mailname

Der Mailname Ihrer Site, z. B. **vbrew.com**.

whoami

Der Name Ihrer Site für News-Zwecke. Häufig wird der UUCP-Sitename, z. B. **vbrew**, verwendet.

explist

Diese Datei sollte die von Ihnen gewünschten Zeiten enthalten, nach denen Artikel bestimmter News-Gruppen ungültig werden. Festplattenplatz spielt hier wahrscheinlich eine entscheidende Rolle.

Um eine erste Hierarchie von News-Gruppen zu erzeugen, besorgen Sie sich die *active*- und *newsgroups*-Datei von der Site, die Sie mit News versorgt und installieren Sie diese in */usr/lib/news*. Sorgen Sie dafür, daß die Dateien **news**gehören und den Modus 644 haben. Entfernen Sie alle *to.**-Gruppen aus der active-Datei und fügen Sie *to.my-site* und *to.feed-site* ein. Auch *junk* und *control* müssen enthalten sein. Die Gruppen *to.** werden normalerweise verwendet, um ihave/sendme-Nachrichten auszutauschen, Sie sollten sie aber auf jeden Fall einrichten, gleichgültig, ob Sie ihave/sendme verwenden wollen oder nicht. Ersetzen Sie nun mit den folgenden Befehlen alle Artikelnummern im zweiten und dritten Feld von *active*:

```
# cp active active.old
# sed 's/ [0-9]* [0-9]* / 0000000000 00001 /' active.old > active
# rm active.old
```

Der zweite Befehl startet *sed*, einen meiner bevorzugten UNIX-Befehle. Der Aufruf ersetzt zwei Strings von Ziffern durch einen String mit Nullen bzw. den String `000001`.

Zum Schluß müssen Sie das News-Spoolverzeichnis und die für die ein- und ausgehenden News benötigten Unterverzeichnisse erzeugen:

```
# cd /var/spool
# mkdir news news/in.coming news/out.going
# chown -R news.news news
# chmod -R 755 news
```

Wenn Sie mit einer älteren Release von C News arbeiten, müssen Sie möglicherweise auch das Verzeichnis *out.master* in Ihrem News-Spoolverzeichnis anlegen.

Wenn Sie einen Newsreader verwenden, der nicht aus C News, sondern aus einer anderen Distribution stammt, als der, aus der Sie Ihr C News haben, ist es möglich, daß einige die News in */usr/spool/news* und nicht in */var/spool/news* spoolen. Falls Ihr Newsreader also keine Artikel finden kann, erzeugen Sie einen symbolischen Link von */usr/spool/news* auf */var/spool/news*.

Nun sind Sie in der Lage, News zu empfangen. Sie müssen übrigens keine anderen als die oben angegebenen Verzeichnisse erzeugen, weil C News jedesmal, wenn es einen Artikel von einer Gruppe empfängt, für die noch kein Verzeichnis existiert, eines anlegt.

Tatsächlich passiert das mit *allen* Gruppen, in die ein Artikel gepostet wurde. Nach einer Weile werden Sie sehen, daß Ihr News-Spoolverzeichnis mit Verzeichnissen für Gruppen wie beispielsweise *alt.lang.teco* vollgepflastert ist, die Sie nie abonniert haben. Sie können dies verhindern, indem Sie alle ungewollten

Gruppen aus *active* entfernen oder regelmäßig ein Shell-Script starten, das alle leeren Verzeichnisse unter */var/spool/news* löscht (natürlich mit Ausnahme von *out.going* und *in.coming*).

C News benötigt einen Benutzer, an den Fehlermeldungen und Statusberichte geschickt werden können. Standardmäßig ist dies der Benutzer **usenet**. Wenn Sie diese Voreinstellung beibehalten, müssen Sie ein oder mehrere Aliases einrichten, damit alle Nachrichten an die entsprechende(n) Person(en) weitergeleitet werden. Sie können aber auch alle Mails an einen anderen Benutzer schicken, indem Sie die Umgebungsvariable `NEWSMASTER` auf den entsprechenden Namen setzen. Dies müssen Sie in der **news**-*crontab* und jedesmal, wenn Sie ein Administrations-Tool von Hand aufrufen, tun. Das Einrichten eines Alias ist also wahrscheinlich einfacher. Lesen Sie dazu Kapitel 14 und Kapitel 15.

Während Sie die Datei */etc/passwd* editieren, sollten Sie sicherstellen, daß für alle Benutzerinnen der echte Name im Feld `pw_gecos` der Paßwort-Datei eingetragen ist (dies ist das vierte Feld). Es ist eine Frage der Usenet-Netiquette, daß der echte Name des Absenders im `From:`-Feld des Artikels erscheint. Wenn Sie einen Mailer eingerichtet haben, haben Sie das wahrscheinlich sowieso schon getan.

Die Datei sys

Die Datei *sys* ist in */usr/lib/news* beheimatet und kontrolliert, welche Hierarchien Sie empfangen und weiterleiten. Obwohl es Verwaltungswerkzeuge wie *addfeed* und *delfeed* gibt, denke ich doch, daß es besser ist, diese Datei von Hand zu verwalten.

Die *sys*-Datei enthält Einträge für jede Site, an die Sie News weiterleiten sowie eine Beschreibung der Gruppen, die Sie akzeptieren. Ein Eintrag sieht folgendermaßen aus:

```
site[/ausschlüsse]:gruppenliste[/distliste][:optionen[:befehle]]
```

Einträge können durch einen Backslash (\) über mehrere Zeilen verteilt werden. Das Doppelkreuz (#) leitet einen Kommentar ein.

site

Der Name der Site, auf die der Eintrag zutrifft. Normalerweise wird der UUCP-Name der Site gewählt. Es muß auch ein Eintrag für Ihre Site in *sys* stehen, weil Sie sonst selbst keine Artikel empfangen können.

Der spezielle Sitename `ME` steht für Ihre Site. Der `ME`-Eintrag definiert alle Gruppen, die Sie lokal speichern wollen. Artikel, die über die `ME`-Zeile nicht zugeordnet werden können, wandern in die *junk*-Gruppe.

Weil C News *`site`* mit den Sitenamen im `Path:`-Headerfeld vergleicht, müssen Sie sicherstellen, daß sie auch stimmen. Manche Sites verwenden den voll qualifizierten Domainnamen oder ein Alias wie `news.`*`site.domain`*. Um zu verhindern, daß Artikel an diese Sites zurückgeliefert werden, müssen Sie eine durch Kommata getrennte Liste erstellen, in der diese Sites ausgeschlossen werden.

Beispielsweise würde der Eintrag für die Site **moria** im *`site`*-Feld den String `moria/moria.orcnet.org` enthalten.

gruppenliste

Eine durch Kommata getrennte Abonnement-Liste aller Gruppen und Hierarchien für diese bestimmte Site. Eine Hierarchie wird durch Angabe des Hierarchie-Präfixes spezifiziert (beispielsweise *comp.os* für alle Gruppen, die mit diesem Präfix beginnen), dem optional das Schlüsselwort *all* folgen kann (z.

B. *comp.os.all*).

Hierarchien oder Gruppen können von der Weiterleitung ausgeschlossen werden, indem ihnen ein Ausrufezeichen vorangestellt wird. Wird eine News-Gruppe mit dieser Liste verglichen, wird immer die größte Übereinstimmung verwendet. Wenn beispielsweise *`gruppenliste`* die folgende Liste enthalten würde:

```
!comp,comp.os.linux,comp.folklore.computers
```

würden keine Gruppen der *comp*-Hierarchie außer *comp.folklore.computers* und alle Gruppen unter *comp.os.linux* an diese Site weitergegeben werden.

Wenn die Site alle News weitergeleitet haben möchte, die auch Sie selbst empfangen, geben Sie einfach `all` als *`gruppenliste`* an.

distliste

Dieser Wert ist von der *`gruppenliste`* durch einen Slash abgetrennt und enthält eine Liste von weiterzuleitenden Distributionen. Auch hier können Sie wieder verschiedene Distributionen durch Voranstellen eines Ausrufezeichens unterbinden. Durch das Schlüsselwort `all` sind alle Distributionen gemeint. Wird keine *`distliste`* angegeben, wird automatisch `all` angenommen.

Beispielsweise könnten Sie die Distributionsliste `all,!local` benutzen, um zu verhindern, daß News, die nur zur lokalen Verwendung bestimmt sind, an andere Sites übertragen werden.

Es gibt normalerweise wenigstens zwei Distributionen: `world`, der Standardverteiler, wenn keiner vom Benutzer spezifiziert wurde, und `local`. Es kann weitere Distributionen geben, die für eine bestimmte Region, Land, Staat etc. gelten. Außerdem gibt es noch zwei Distributionen, die nur von C News verwendet werden. Diese sind `sendme` und `ihave`, die für das sendme/ihave-Protokoll Anwendung finden.

Die Verwendung von Distributionen ist Thema zahlreicher Debatten. Zum einen erzeugen manche Newsreader wirre Distributionen, indem sie einfach Top-Level-Hierarchien wie beispielsweise *comp* verwenden, wenn sie in *comp.os.linux* posten. Auch Distributionen für bestimmte Regionen sind häufig fragwürdig, weil News auch außerhalb Ihrer Region wandern können, wenn sie über das Internet geschickt werden.(3) Distributionen, die sich dagegen an eine Organisation wenden, sind durchaus sinnvoll; so gilt es beispielsweise zu verhindern, daß vertrauliche Informationen ein Unternehmens-Netzwerk verlassen. Dieses Ziel wird allerdings besser erreicht, indem eine separate News-Gruppe oder Hierarchie angelegt wird.

optionen

Hiermit beschreiben Sie verschiedene Parameter für den Feed. Dieses Feld kann leer sein oder eine Kombination der folgenden Schlüsselwörter enthalten:

`F`

Diese Option aktiviert den Batch-Betrieb.

`f`

Diese Option ist nahezu identisch mit `F`, erlaubt es C News aber, die Größe der ausgehenden Batches präziser zu berechnen.

`I`

Mit dieser Option erzeugt C News eine Artikelliste, die von ihave/sendme verwendet werden kann. Weitere Modifikationen werden in den Dateien *sys* und *batchparms* benötigt, um

ihave/sendme zu aktivieren.

`n`

Dies erzeugt Batch-Dateien für die aktive NNTP-Übertragung wie z. B. mit *nntpxmit* (siehe Kapitel 18, *Eine Einführung in NNTP*). Die Batch-Dateien enthalten den Dateinamen des Artikels zusammen mit der Message-ID.

`L`

Damit weisen Sie C News an, nur Dateien zu übertragen, die auf Ihrer Site gepostet wurden. Dieser Option kann eine Dezimalzahl *n* folgen, die dafür sorgt, daß C News Artikel nur dann überträgt, wenn diese innerhalb von *n* Hops von Ihrer Site entfernt gepostet wurden. C News ermittelt die Anzahl der Hops aus dem `Path:`-Feld.

`u`

Weist C News an, Batches nur von Artikeln aus nicht moderierten Gruppen zu erstellen.

`m`

Weist C News an, Batches nur von Artikeln aus moderierten Gruppen zu erstellen.

Sie dürfen maximal eine der Optionen `F`, `f`, `I` oder `n` benutzen.

befehle

Dieses Feld enthält einen Befehl, der bei jedem Artikel ausgeführt wird, solange der Batch-Betrieb nicht aktiviert ist. Der Artikel wird über die Standardeingabe übergeben. Diese Möglichkeit sollte nur bei sehr kleinen Feeds verwendet werden, weil sonst die Last auf beiden Systemen zu hoch wird.

Der Standardbefehl lautet:

```
uux -- -r -z remote-system!rnews
```

Damit wird *rnews* auf dem entfernten System gestartet und der Artikel über die Standardeingabe übergeben.

Der für in diesem Feld angegebene Befehle verwendete Standard-Suchpfad ist */bin:/usr/bin:-/usr/lib/news/bin/batch*. Das letzte Verzeichnis enthält eine Reihe von Shell-Scripten, deren Namen mit *via* beginnen. Diese werden später in diesem Kapitel noch kurz besprochen.

Ist der Batch-Betrieb über eine der Optionen `F`, `f`, `I` oder `n` aktiviert worden, erwartet C News in diesem Feld einen Dateinamen anstelle eines Befehls. Beginnt der Dateiname nicht mit einem Slash (/), wird angenommen, daß er relativ zu */var/spool/news/out.going* ist. Ist das Feld leer, wird standardmäßig auf *remote-system/togo* zurückgegriffen.

Wenn Sie C News einrichten, müssen Sie höchstwahrscheinlich Ihre eigene *sys*-Datei schreiben. Um Ihnen die Sache zu erleichtern, finden Sie nachfolgend eine Beispieldatei für **vbrew.com**, von der Sie übernehmen können, was Sie benötigen.

```
# Wir nehmen alles, was man uns gibt.
ME:all/all::
# Wir senden alles, was wir empfangen, an moria, außer lokalen und
# Brauerei-internen Artikeln. Wir arbeiten im Batch-Betrieb.
moria/moria.orcnet.org:all,!to,to.moria/all,!local,!brewery:f:
# Wir übertragen comp.risks über Mail an jack@ponderosa.uucp.
```

```
ponderosa:comp.risks/all::rmail jack@ponderosa.uucp
# swim bekommt einen kleinen Feed.
swim/swim.twobirds.com:comp.os.linux,rec.humor.oracle/all,!local:f:
# Mail-Map-Artikel werden für spätere Bearbeitung gespeichert.
usenet-maps:comp.mail.maps/all:F:/var/spool/uumaps/work/batch
```

Die Datei active

Die Datei *active* ist unter */usr/lib/news* zu finden und enthält eine Liste aller Ihrer Site bekannten Gruppen sowie aller Artikel, die momentan online sind. Sie werden diese Datei nur selten bearbeiten müssen, aber wir erklären sie an dieser Stelle der Vollständigkeit halber. Einträge haben die folgende Form:

```
newsgruppe Obergrenze Untergrenze rechte
```

newsgruppe ist der Gruppenname. `Untergrenze` und `Obergrenze` stehen für die niedrigste bzw. höchste Nummer von Artikeln, die gerade verfügbar sind. Sind gerade keine Artikel vorhanden, ist `Untergrenze` gleich `Obergrenze`+1.

Zumindest ist das so vorgesehen. Allerdings wird dieses Feld aus Effizienzgründen von C News nicht aktualisiert. Das wäre wiederum kein so großer Verlust, wenn nicht einige Newsreader von diesem Wert abhängig wären. Beispielsweise überprüft *trn* dieses Feld, um zu sehen, ob es irgendwelche Artikel aus der Thread-Datenbank löschen kann. Um das `Untergrenze`-Feld zu aktualisieren, müssen Sie daher regelmäßig den Befehl *updatemin* (bei älteren Versionen von C News das *upact*-Script) ausführen.

`rechte` ist ein Parameter, der detailliert beschreibt, welche Zugriffsrechte Benutzern bei dieser Gruppe zustehen. Er hat einen der folgenden Werte:

y

Benutzer dürfen in diese Gruppe posten.

n

Benutzer dürfen nicht in diese Gruppe posten. Die Artikel dieser Gruppe dürfen aber trotzdem gelesen werden.

x

Diese Gruppe ist lokal deaktiviert worden. Dies kommt manchmal vor, wenn News-Administratoren (oder deren Vorgesetzte) weltanschauliche Vorbehalte gegen bestimmten Gruppen haben.

Artikel, die für diese Gruppe empfangen werden, werden nicht lokal gespeichert, obwohl sie an die Sites weitergegeben werden, die sie angefordert haben.

m

Bezeichnet eine moderierte Gruppe. Versucht ein Benutzer, eine Nachricht in diese Gruppe zu posten, wird ein intelligenter Newsreader ihm einen Hinweis geben und den Artikel statt dessen an den Moderator schicken. Die Adresse des Moderators wird aus der Datei *moderators* in */usr/lib/news* gelesen.

=reale-gruppe

Kennzeichnet `newsgruppe` als lokales Alias für die Gruppe `reale-gruppe`. Alle Artikel, die in `newsgruppe` gepostet werden, werden in diese Gruppe umgeleitet.

Bei C News müssen Sie auf diese Datei in der Regel nicht direkt zugreifen. Gruppen können über die Befehle

addgroup und *delgroup* lokal erzeugt und gelöscht werden (siehe den nachfolgenden Abschnitt, »*Verwaltungsaufgaben und -Tools*«). Die Control-Message `newgroup` erzeugt eine Gruppe im gesamten Usenet, während die Control-Message `rmgroup` eine Gruppe löscht. *Senden Sie eine solche Nachricht niemals selbst!* Anweisungen zum Anlegen einer neuen News-Gruppe finden Sie in den monatlichen Postings in *news.announce.newusers*.

Eine sehr eng mit *active* verknüpfte Datei ist *active.times*. Jedesmal wenn eine Gruppe angelegt wird, speichert C News eine Notiz in dieser Datei, die das Datum der Erzeugung enthält, ob dies über eine `newgroup`-Control-Message oder lokal erfolgte, und wer sie erzeugte. Diese Informationen können dann von Newsreadern verwendet werden, die die Benutzer über neu erzeugte Gruppen informieren wollen. Außerdem wird sie vom NNTP-Befehl `NEWGROUPS` genutzt.

Stapelverarbeitung von Artikeln (Batching)

News-Batches verwenden ein bestimmtes Format, das für B News, C News und INN identisch ist. Jedem Artikel wird eine Zeile wie die folgende vorangestellt:

```
#! rnews größe
```

größe bezeichnet die Anzahl Bytes in diesem Artikel. Bei Batch-Kompression wird die Datei als Ganzes komprimiert und eine weitere Zeile vorangestellt, die angibt, welches Programm zum Entpacken verwendet werden soll. Das Standard-Kompressionstool ist *compress*, was durch folgende Zeile gekennzeichnet wird:

```
#! cunbatch
```

Manchmal, wenn Batches über E-Mail-Software übertragen werden, die das achte Bit von allen Daten entfernt, wird ein komprimierter Batch durch die sogenannte *c7-Kodierung*; geschützt. Diese Batches sind durch *c7unbatch* markiert.

Wird ein Batch an *rnews* auf einer anderen Site übergeben, werden diese Markierungen geprüft und der Stapel wird entsprechend korrekt verarbeitet. Manche Sites verwenden auch andere Kompressions-Tools wie *gzip* und stellen den so komprimierten Daten *zunbatch* voran. C News kann mit solchen Headern nicht umgehen, da diese dem Standard nicht entsprechen. Sie müssen die Quelltexte entsprechend modifizieren, um sie zu unterstützen.

Bei C News wird der Batch-Betrieb über */usr/lib/news/bin/batch/sendbatches* abgewickelt. Dabei wird eine Liste von Artikeln aus der Datei *site/togo* gelesen, aus denen dann mehrere News-Batches erzeugt werden. Der Befehl sollte einmal in der Stunde oder sogar noch häufiger ausgeführt werden, wenn Sie ein hohes Datenaufkommen haben. Kontrolliert wird der Batch-Betrieb über die Datei *batchparms* in */usr/lib/news*. Diese Datei beschreibt die maximal erlaubte Batch-Größe für jede Site, das zur Stapelverarbeitung und optional zur Komprimierung zu verwendende Programm sowie die zur Auslieferung zu benutzende Transportart. Sie können Batch-Parameter für jede einzelne Site einstellen, aber auch Standardwerte für Sites vorgeben, die nicht explizit aufgeführt sind.

Um die Stapelverarbeitung für eine bestimmte Site durchzuführen, müssen Sie den folgenden Befehl verwenden:

```
# su news -c "/usr/lib/news/bin/batch/sendbatches site"
```

Ohne Angabe von Argumenten verarbeitet *sendbatches* alle Batch-Queues. Wie das Wort »alle« interpretiert

wird, hängt von der Präsenz eines Standardeintrags in *batchparms* ab. Ist einer vorhanden, werden alle Verzeichnisse in */var/spool/news /out.going* geprüft. Andernfalls arbeitet es sich durch alle Einträge der *batchparms* hindurch. Sucht *sendbatches* das Verzeichnis *out.going* ab, werden nur solche Verzeichnisse als Sitenamen interpretiert, die keine Punkte oder »at«-Zeichen (`@`) enthalten.

Wenn Sie C News installieren, finden Sie höchstwahrscheinlich eine Version von *batchparms* in Ihrer Distribution, die einen sinnvollen Standardeintrag enthält, so daß gute Aussichten bestehen, daß Sie diese Datei gar nicht anfassen müssen. Falls Sie es doch tun müssen, beschreiben wir das Format dieser Datei. Jede Zeile besteht aus sechs Feldern, die durch Leerzeichen oder Tabulatoren voneinander getrennt sind:

`site größe max batcher muncher transport`

site ist der Name der Site, für die dieser Eintrag gilt. Die Datei *togo* für diese Site muß in *out.going/togo* im Spool-Verzeichnis zu finden sein. Der Site-Name `/default/` bezeichnet den Standardeintrag.

`größe` ist die maximale Größe von erzeugten Artikel-Batches (vor der Komprimierung). Bei einzelnen Artikeln, die größer als dieser Wert sind, macht C News eine Ausnahme und erzeugt einen einzelnen Batch.

`max` enthält die maximale Zahl der erzeugten und zum Transfer angewiesenen Batches, bevor der Batch-Betrieb für diese Site vorübergehend eingestellt wird. Das ist dann sinnvoll, wenn eine andere Site für lange Zeit heruntergefahren sein sollte, weil auf diese Weise verhindert wird, daß C News Ihre UUCP-Spoolverzeichnisse mit Abermillionen von News-Batches übersät.

C News ermittelt die Anzahl wartender Batches über das *queulen*-Script in */usr/lib /news/bin*. Vince Skahans *newspak*-Release sollte ein Script für BNU-kompatible UUCP-Versionen enthalten. Wenn Sie mit einer anderen Art von Spool-Verzeichnissen arbeiten, beispielsweise bei Taylor-UUCP, müssen Sie Ihr eigenes Script schreiben.(4)

Das Feld *`batcher`* enthält den Befehl, der verwendet wird, um aus der Liste von Artikeln in der *togo*-Datei einen Batch zu erzeugen. Bei normalen Feeds ist dies üblicherweise *batcher*. Für andere Zwecke können alternative Batcher verwendet werden. Beispielsweise verlangt das ihave/sendme-Protokoll, daß die Artikelliste in ihave- oder sendme-Control-Messages umgewandelt wird, die in die News-Gruppe *to.site* gepostet werden. Dies wird durch *batchih* und *batchsm* erledigt.

Das Feld *`muncher`* enthält den Befehl, der zur Komprimierung verwendet wird. Üblicherweise ist dies *compcun*, ein Script, das einen komprimierten Batch erzeugt.(5) Alternativ könnten Sie auch ein Script verwenden, das *gzip* verwendet, beispielsweise *gzipcun* (um es deutlich zu sagen: Sie müssen es selbst schreiben). Sie müssen sicherstellen, daß *uncompress* auf dem anderen Rechner so gepatcht ist, daß es mit *gzip* komprimierte Dateien erkennt.

Falls die andere Site nicht über einen *uncompress*-Befehl verfügt, können Sie *nocomp* angeben. In diesem Fall wird keine Komprimierung durchgeführt.

Das letzte Feld, *`transport`*, beschreibt die zu verwendende Transportart. Eine Reihe von Standardbefehlen, deren Namen mit *via* beginnen, stehen für verschiedene Transportarten bereit. *sendbatches* übergibt ihnen den Namen der Ziel-Site in der Kommandozeile. Beim `/default/`-Eintrag generiert es den Site-Namen aus dem *`site`*-Feld, indem es alle Zeichen ab dem ersten Punkt oder Slash abschneidet. Beim Eintrag `/default/` werden statt dessen die Verzeichnisnamen aus *out.going* verwendet.

Die beiden Befehle *viauux* und *viauuxz* nutzen *uux*, um *rnews* auf dem anderen System auszuführen Der zweite setzt dabei die Option -z (für ältere Versionen) von *uux*. Damit wird verhindert, daß für jeden übertragenen Artikel Erfolgsmeldungen zurückgeliefert werden. Ein anderer Befehl, *viamail*, verschickt Artikel-Batches per Mail an den Benutzer **rnews** des entfernten Systems. Natürlich ist es bei dieser Methode notwendig, daß das entfernte System die gesamte Post für **rnews** an das lokale News-System weitergibt. Eine komplette Liste der möglichen Transportarten finden Sie in der *newsbatch*-Manpage.

Alle Befehle in den letzten drei Feldern müssen entweder in *out.going/site* oder in */usr/lib/news/bin/batch* zu finden sein. Bei den meisten handelt es sich um Scripten, die Sie einfach um neue Tools für Ihre persönlichen Bedürfnisse erweitern können. Aufgerufen werden sie über eine Pipe. Die Liste der Artikel wird dem Batcher über die Standardeingabe übergeben, der daraus den Batch erzeugt und an die Standardausgabe weiterleitet. Diese wird über eine Pipe an den Muncher weitergeleitet und so weiter.

Eine Beispieldatei ist nachfolgend aufgeführt.

```
# batchparms-Datei für die Brauerei
# site        | größe  |max    |batcher  |muncher    |transport
#-------------+--------+-------+---------+-----------+-----------
/default/       100000  22      batcher   compcun     viauux
swim             10000  10      batcher   nocomp      viauux
```

News und Expiring

Bei B News wurden alte Artikel mit Hilfe des Programms *expire* entfernt. Als Argumente wurden eine Liste von News-Gruppen sowie eine Zeitspanne übergeben, nach denen Artikel gelöscht werden sollten. Um verschiedene Hierarchien unterschiedlich lange vorzuhalten, mußte man ein Script schreiben, das *expire* für jede Hierarchie einzeln aufrief. C News bietet Ihnen hierfür eine bequemere Lösung. In einer Datei namens *explist* können Sie News-Gruppen zusammen mit den Zeitintervallen, nach denen jeweils gelöscht werden soll, festhalten. Ein Befehl namens *doexpire* wird dann üblicherweise einmal pro Tag von *cron* aus gestartet, der dann alle Gruppen entsprechend der Liste bereinigt. Gelegentlich möchten Sie Artikel aus verschiedenen Gruppen behalten, selbst wenn die Ablaufzeit bereits überschritten wurde. Beispielsweise könnten Sie alle Programme aus *comp.sources.unix* aufheben wollen. Dies wird als *Archivierung* bezeichnet. *explist* erlaubt Ihnen die Markierung von zur Archivierung bestimmten Gruppen.

Ein Eintrag in *explist* hat das folgende Format:

```
gruppenliste rechte zeiten archiv
```

gruppenliste ist eine durch Kommata unterteilte Liste der News-Gruppen, für die dieser Eintrag gilt. Hierarchien können durch das Gruppenpräfix spezifiziert werden, dem optional das Schlüsselwort `all` folgen kann. So können Sie etwa für einen Eintrag, der für alle Gruppen unter *comp.os* gilt, sowohl `comp.os` als auch `comp.os.all` angeben.

Werden Artikel aus einer Gruppe gelöscht, wird der Name nacheinander mit allen Einträgen in *explist* verglichen. Der erste passende Eintrag wird dann verwendet. Um beispielsweise den größten Teil von *comp* nach vier Tagen zu löschen, mit Ausnahme von *comp.os.linux.announce*, die erst nach einer Woche entfernt werden sollen, müssen Sie einfach einen Eintrag mit einer Frist von sieben Tagen für letztere definieren, dem der Eintrag von *comp* folgt, bei dem nur vier Tage angegeben werden.

Das Feld `rechte` legt fest, ob der Eintrag für moderierte, unmoderierte oder beliebige Gruppen gilt. Dieses Feld kann einen der Werte `m`, `u` oder `x` annehmen, was entsprechend moderiert, unmoderiert oder beliebig bedeutet.

Das dritte Feld, `zeiten`, enthält üblicherweise nur eine Zahl. Die Zahl bestimmt die Anzahl von Tagen, nach denen ein Artikel für ungültig erklärt wird. Diese Ablaufzeit kann durch eine explizite Ablaufzeit im Feld `Expires:` des Artikel-Headers überschrieben werden. Beachten Sie, daß diese Zahl die Ablaufzeit nach dem *Eintreffen* auf Ihrer Site bestimmt und sich nicht an der Zeit des Postings orientiert.

Das `zeiten`-Feld kann aber auch komplexer sein. Es kann eine Kombination von bis zu drei Zahlen enthalten, die voneinander durch einen Bindestrich getrennt sind. Die erste Zahl bestimmt die Anzahl von Tagen, die ein Artikel mindestens erhalten bleibt. Es ist kaum sinnvoll, diesen Wert auf etwas anderes all null zu setzen. Das zweite Feld enthält die bereits oben beschriebene Zahl von Tagen, nach denen ein Artikel gelöscht wird. Der dritte Wert bestimmt die bedingungslose Ablaufzeit eines Artikels, gleichgültig, ob ein `Expires:`-Feld existiert oder nicht. Wird nur der mittlere Wert angegeben, werden für die beiden anderen Standardwerte eingesetzt. Diese können mit dem Spezialeintrag `/bounds/` bestimmt werden, der später noch beschrieben wird.

Im vierten Feld, `archiv`, wird festgelegt, ob die News-Gruppe archiviert wird und wenn ja, wo. Wird keine Archivierung gewünscht, sollte an dieser Stelle ein Bindestrich stehen. Wird die Archivierung gewünscht, steht an dieser Stelle der volle Pfadname (der auf ein Verzeichnis zeigt) oder ein at-Symbol (@). Das at-Symbol steht für ein Standard-Archivierungsverzeichnis, das dann an *doexpire* über die Option *-a* in der Kommandozeile übergeben werden muß. Ein Archiv-Verzeichnis sollte **news** gehören. Archiviert *doexpire* etwa einen Artikel aus *comp.sources.unix*, speichert es diesen im Verzeichnis *comp/sources/unix* unter dem Archiv-Verzeichnis. Wenn dieses noch nicht existiert, wird es automatisch erzeugt. Das Archiv-Verzeichnis selbst wird allerdings nicht automatisch erzeugt.

In der Datei *explist* gibt es zwei spezielle Einträge, auf die sich *doexpire* verläßt. Anstelle einer Liste mit News-Gruppen verwenden sie die Schlüsselwörter `/bounds/` und `/expired/`. Der `/bounds/`-Eintrag enthält die Standardwerte der drei oben beschriebenen Einträge für das `zeiten`-Feld.

Im `/expired/`-Feld wird festgelegt, wie lange C News an Zeilen aus der *history*-Datei festhält. Dies ist notwendig, weil C News Zeilen aus der history-Datei nicht automatisch entfernt, wenn der oder die entsprechenden Artikel gelöscht wurden. Statt dessen wird die Zeile behalten, falls noch eine Kopie des Artikels nach diesem Datum eintreffen sollte. Falls Sie von nur einer Site mit News versorgt werden, können Sie diesen Wert klein halten. In UUCP-Netzen ist es ratsam, ihn auf ein paar Wochen zu setzen, abhängig von den Verzögerungen, mit denen News von diesen Sites bei Ihnen eintreffen.

Ein Beispiel einer solchen *explist*-Datei mit ziemlich kurzen Zeitintervallen ist nachfolgend aufgeführt.

```
# history-Zeilen zwei Wochen halten. Niemand bekommt mehr als drei Monate.
/expired/                       x       14      -
/bounds/                        x       0-1-90  -
# Gruppen, die wir länger halten wollen als den Rest.
comp.os.linux.announce          m       10      -
comp.os.linux                   x       5       -
alt.folklore.computers          u       10      -
rec.humor.oracle                m       10      -
soc.feminism                    m       10      -
# Archiviere die *.sources-Gruppen
```

```
comp.sources,alt.sources        x       5       @
# Standardwerte für technische Gruppen
comp,sci                        x       7       -
# Genug für ein langes Wochenende
misc,talk                       x       4       -
# Schnell weg mit überflüssigem Ballast
junk                            x       1       -
# Steuernachrichten sind auch nicht von besonderem Interesse
control                         x       1       -
# Eintrag, der den ganzen Rest abfängt
all                             x       2       -
```

Das Löschen von Artikeln zieht einige mögliche Probleme nach sich. Eines dieser Probleme ist, daß Ihr Newsreader sich möglicherweise auf das dritte Feld der active-Datei verläßt, das die kleinste Nummer des online verfügbaren Artikels enthält. Werden Artikel gelöscht, aktualisiert C News dieses Feld nicht. Wenn dieses Feld die wirkliche Situation widerspiegeln muß (oder soll), muß jedesmal ein Programm names *updatemin* ausgeführt werden, nachdem *doexpire*aufgerufen wurde. (Bei älteren C News-Versionen wurde ein Script namens *upact* verwendet.)

Sollen unter C News Artikel gelöscht werden, werden nicht die Verzeichnisse der News-Gruppen durchsucht, sondern es wird einfach anhand der *history*-Datei überprüft, ob ein Artikel das Ablaufdatum überschritten hat.(6) Ist Ihre history-Datei aus irgendeinem Grund nicht auf dem neuesten Stand, könnten Artikel für immer auf Ihrer Festplatte schmoren, weil C News sie einfach vergessen hat.(7) Sie können die Synchronisation durch das Script *addmissing* wiederherstellen, das in */usr/lib /news/bin/maint* zu finden ist und fehlende Artikel in die *history* aufnimmt. Eine andere Möglichkeit bietet *mkhistory*, das die gesamte Datei neu erstellt. Denken Sie daran, diese Operationen als **news** durchzuführen, weil Sie ansonsten eine *history*-Datei erzeugen, die von C News nicht gelesen werden kann.

Weitere Dateien

Es gibt eine Reihe von Dateien, die das Verhalten von C News kontrollieren, für den Betrieb aber nicht von grundlegender Bedeutung sind. Alle diese Dateien sind in */usr/lib/news* zu finden und werden nachfolgend kurz beschrieben.

newsgroups

Eine mit *active* verwandte Datei, die eine Liste mit dem Namen jeder News-Gruppe zusammen mit einer einzeiligen Beschreibung des Hauptthemas enthält. Diese Datei wird automatisch aktualisiert, wenn C News die Steuernachricht `checknews` empfängt.

localgroups

Wenn Sie eine Reihe lokaler Gruppen besitzen, über die sich C News nicht jedesmal beschweren soll, wenn Sie eine `checknews`-Nachricht empfangen, dann tragen Sie ihre Namen und die Beschreibungen in diese Datei ein. Das Format entspricht dem von *newsgroups*.

mailpaths

Diese Datei enthält die Adresse des Moderators jeder moderierten Gruppe. Jede Zeile besteht aus dem Namen der Gruppe, dem, durch einen Tabulator getrennt, die E-Mail-Adresse des Moderators folgt.

Zwei spezielle Einträge sind standardmäßig bereits enthalten: `backbone` und `internet`. Beide enthalten jeweils in Bang Path-Notation den Pfad zur nächsten Backbone-Site bzw. zur nächsten Site,

die die RFC822-Adressierung (*benutzer@host*) versteht. Die Standardeinträge lauten:

```
internet    %S
backbone    %S
```

Sie müssen den Eintrag `internet` nicht ändern, wenn Sie *smail* oder *sendmail* installiert haben, weil diese die RFC822-Adressierung verstehen.

Der `backbone`-Eintrag wird immer verwendet, wenn ein Benutzer an eine moderierte Gruppe postet, die nicht explizit aufgeführt ist. Ist beispielsweise der Name der News-Gruppe *alt.sewer* und enthält der `backbone`-Eintrag den Wert *path!%s*, schickt C News den Artikel per E-Mail an *path!alt-sewer* in der Hoffnung, daß die Backbone-Maschine in der Lage ist, den Artikel weiterzuleiten. Um herauszufinden, welchen Pfad Sie benutzen müssen, wenden Sie sich an den News-Administrator der Site, die Sie mit News versorgt. Als letzte Möglichkeit können Sie auch *uunet.uu.net!%s* verwenden.

distributions

Diese Datei ist eigentlich keine C News-Datei, sondern wird von einigen Newsreadern und *nntpd* benutzt. Enthalten ist eine Liste der von Ihrer Site erkannten Distributionen sowie eine Beschreibung ihrer (gewünschten) Effekte.

So besitzt die virtuelle Brauerei beispielsweise die folgende Datei:

```
world         überall auf der Welt
local         nur lokal für diese Site
nl            nur die Niederlande
mugnet        nur MUGNET
fr            nur Frankreich
de            nur Deutschland
brewery       nur virtuelle Brauerei
```

log

In dieser Datei werden alle Aktivitäten von C News festgehalten. Sie sollte regelmäßig über *newsdaily* gestutzt werden. Kopien alter Log-Dateien werden unter den Namen *log.o*, *log.oo* und so weiter gespeichert.

errlog

In dieser Datei werden alle von C News erzeugten Fehlermeldungen gespeichert. Artikel, die aufgrund einer falschen Gruppe aussortiert wurden und ähnliches fallen aber nicht hierunter. Diese Datei wird von *newsdaily* automatisch per E-Mail an den Newsmaster (standardmäßig usenet) geschickt, wenn sie nicht leer ist.

errlog

wird von *newsdaily* gelöscht. Alte Kopien werden in *errlog.o* etc. gehalten.

batchlog

In dieser Datei werden alle Durchläufe von *sendbatches* gespeichert. Diese Datei ist üblicherweise nur von geringem Interesse. Wird auch von *newsdaily* benutzt.

watchtime

Eine leere Datei, die jedesmal erzeugt wird, wenn *newswatch* gestartet wird.

Steuermeldungen

Das Protokoll der Usenet-News kennt eine spezielle Kategorie von Artikeln, die bestimmte Antworten bzw. Aktionen durch das News-System erzwingen. Diese werden als »Steuermeldungen« (*control messages*) bezeichnet. Erkannt werden sie durch das Vorhandensein eines `Control:`-Feldes im Artikel-Header, das den Namen der auszuführenden Steuerungsoperation enthält. Es gibt die verschiedensten Operationen, die alle durch Shell-Scripten übernommen werden, die in */usr/lib/news/ctl* zu finden sind.

Die meisten dieser Aktionen werden automatisch ausgeführt, während C News den Artikel bearbeitet, wobei der Newsmaster nicht darüber informiert wird. Standardmäßig werden nur `checkgroups`-Meldungen an den Newsmaster weitergeleitet, was Sie aber durch Editieren der Scripten anpassen können.

Die cancel-Meldung

Die bekannteste Meldung ist `cancel`. Damit kann ein Benutzer einen vorher geposteten Artikel löschen (aufheben). Diese Nachricht entfernt den Artikel, wenn er existiert, aus den Spool-Verzeichnissen. Die `cancel`-Meldung wird an alle Sites weitergeleitet, die News aus den Gruppen beziehen, die diese Meldung betrifft. Dabei ist es gleichgültig, ob die Site den Artikel bereits gesehen hat oder nicht. So wird die Möglichkeit in Betracht gezogen, daß der eigentliche Artikel erst nach der cancel-Meldung eintrifft. Einige News-Systeme erlauben es, Nachrichten anderer Personen zu canceln. Das sollten Sie natürlich niemals tun.

newgroup und rmgroup

Die zwei Steuermeldungen, die dem Erzeugen und Entfernen von News-Gruppen dienen, sind `newgroup` und `rmgroup`. News-Gruppen unter den »üblichen« Hierarchien können nur nach einer Diskussion und einer abschließenden Abstimmung zwischen Usenet-Lesern erzeugt werden. Die bei der *alt*-Hierarchie gültigen Regeln kommen der totalen Anarchie recht nahe. Weitere Informationen finden Sie in den regelmäßigen Veröffentlichungen in *news.announce.newusers* und *news.announce.newgroups*. Senden Sie niemals selbst eine `newgroup`- oder `rmgroup`-Nachricht, solange Sie nicht ganz sicher sind, daß Sie das auch wirklich dürfen.

Die checkgroups-Meldung

`checkgroups`-Meldungen werden von News-Administratoren verschickt, damit alle Sites ihre *active*-Dateien mit den Gegebenheiten im Usenet synchronisieren können. So könnten etwa kommerzielle Internet-Service-Provider eine solche Nachricht an die Sites ihrer Kunden schicken. Einmal im Monat wird die »offizielle« `checkgroups`-Meldung für die Haupt-Hierarchien vom entsprechenden Moderator in *comp.announce.newgroups* gepostet. Allerdings wird sie als normaler Artikel gepostet und nicht als Steuermeldung. Um die `checkgroups`-Operation durchzuführen, speichern Sie den Artikel in einer Datei, etwa */tmp/check*, löschen alles bis zum Beginn der eigentlichen Steuermeldung und übergeben es mit dem folgenden Befehl an das `checkgroups`-Script:

```
# su news -c "/usr/lib/news/bin/ctl/checkgroups" < /tmp/check
```

Dadurch wird Ihre *newsgroups*-Datei aktualisiert, wobei die in *localgroups* aufgeführten Gruppen hinzugefügt werden. Die alte *newsgroups*-Datei wird unter dem Namen *newsgroups.bac* gespeichert. Das lokale Posten einer solchen Meldung funktioniert übrigens selten, weil *inews* einen so großen Artikel nicht akzeptiert.

Erkennt C News Unterschiede zwischen der `checkgroups`-Liste und der *active*-Datei, erzeugt es eine Liste von Befehlen, die Ihre Site auf den neuesten Stand bringen, und schickt sie an den News-Administrator.

Die Ausgabe sieht üblicherweise wie folgt aus:

```
From news Sun Jan 30 16:18:11 1994
Date: Sun, 30 Jan 94 16:18 MET
From: news (News Subsystem)
To: usenet
Subject: Problems with your active file
The following newsgroups are not valid and should be removed.
        alt.ascii-art
        bionet.molbio.gene-org
        comp.windows.x.intrisics
        de.answers
You can do this by executing the commands:
        /usr/lib/news/bin/maint/delgroup alt.ascii-art
        /usr/lib/news/bin/maint/delgroup bionet.molbio.gene-org
        /usr/lib/news/bin/maint/delgroup comp.windows.x.intrisics
        /usr/lib/news/bin/maint/delgroup de.answers
The following newsgroups were missing.
        comp.binaries.cbm
        comp.databases.rdb
        comp.os.geos
        comp.os.qnx
        comp.unix.user-friendly
        misc.legal.moderated
        news.newsites
        soc.culture.scientists
        talk.politics.crypto
        talk.politics.tibet
```

Wenn Sie eine solche Meldung von Ihrem News-System erhalten, sollten Sie ihr aber nicht blind vertrauen. Je nachdem, wer Ihnen die `checkgroups`-Meldung geschickt hat, können einige Gruppen oder sogar ganze Hierarchien fehlen. Seien Sie also äußerst vorsichtig beim Entfernen jedweder Gruppen. Wenn Sie Gruppen finden, die als fehlend aufgeführt sind, die Sie aber auf Ihrer Site haben möchten, müssen Sie diese über das Script *addgroup* aufnehmen. Speichern Sie die Liste der fehlenden Gruppen in einer Datei und übergeben Sie sie an das folgende kleine Script:

```
#!/bin/sh
cd /usr/lib/news
while read group; do
    if grep -si "^$group[[:space:]].*moderated" newsgroup; then
        mod=m
    else
        mod=y
    fi
    /usr/lib/news/bin/maint/addgroup $group $mod
done
```

sendsys, version und senduuname

Zum Schluß gibt es noch drei Meldungen, mit denen Sie etwas über die Netzwerk-Topologie erfahren können. Dies sind `sendsys`, `version` und `senduuname`. Daraufhin überträgt C News die *sys*-Datei an den Absender, die aus einem String besteht, der die Software-Version widerspiegelt bzw. die Ausgabe von *uuname*. Bei der `version`-Meldung gibt sich C News dagegen eher wortkarg und liefert ein einfaches, schmuckloses »C« zurück.

Auch in diesem Fall sollten Sie *niemals* eine solche Meldung verschicken, solange Sie nicht sichergestellt haben, daß sie Ihr (regionales) Netzwerk nicht verlassen kann. Antworten auf `sendsys`-Meldungen können ein UUCP-Netzwerk nämlich ganz schnell in die Knie zwingen.(8)

C News in einer NFS-Umgebung

Ein einfacher Weg, News in einem lokalen Netzwerk zu verteilen, besteht darin, alle News auf einem zentralen Host zu speichern und die relevanten Verzeichnisse über NFS zu exportieren, so daß die Newsreader die Artikel direkt von dort lesen können. Der Vorteil dieser Methode gegenüber NNTP liegt darin, daß der durch den Abruf und das Threading verursachte Overhead deutlich geringer ist. Andererseits gewinnt NNTP in heterogenen Netzwerken, wo sich das Equipment der einzelnen Hosts stark unterscheidet oder wo Benutzer keine gleichwertigen Accounts auf der Server-Maschine besitzen.

Bei NFS müssen auf einem lokalen Host gepostete Artikel an die zentrale Maschine weitergeleitet werden, weil der gleichzeitige Zugriff auf administrative Dateien wie active zu Inkonsistenzen führen kann. Darüber hinaus wollen Sie eventuell auch Ihren Spool-Bereich schützen, indem Sie ihn nur mit Leserechten exportieren, was wiederum die Weiterleitung an die zentrale Maschine erfordert.

C News verwaltet dies transparent. Wenn Sie einen Artikel posten, startet Ihr Newsreader üblicherweise *inews*, um den Artikel an das News-System zu übergeben. Dieser Befehl führt eine ganze Reihe von Prüfungen des Artikels durch, vervollständigt den Header und prüft die Datei *server* in */usr/lib/news*. Wenn die Datei existiert und der Hostname sich von dem des lokalen Host unterscheidet, wird *inews* über *rsh* auf dem Server-Host ausgeführt. Weil das *inews*-Script eine Reihe binärer Programme und Support-Dateien von C News verwendet, müssen Sie C News entweder lokal installiert haben oder die News-Software über den Server mounten.

Damit der *rsh*-Aufruf ordnungsgemäß funktionieren kann, muß der Benutzer einen entsprechenden Account auf dem Server-System besitzen, d. h. einen, bei dem nicht nach einem Paßwort gefragt wird.

Stellen Sie sicher, daß der in *server* stehende Hostname exakt mit der Ausgabe des *hostname*-Befehls auf der Server-Maschine übereinstimmt, weil C News sonst beim Versuch, den Artikel auszuliefern, in einer Endlosschleife hängenbleibt.

Verwaltungsaufgaben und -Tools

Trotz der Komplexität von C News kann das Leben eines News-Administrators recht einfach sein, weil C News eine breite Palette von Verwaltungs-Tools bereitstellt. Einige wie beispielsweise *newsdaily*sind für den regelmäßigen Betrieb aus *cron* heraus gedacht. Die Verwendung dieser Skripten reduziert die Anforderungen an die tägliche Pflege Ihrer C News-Installation ganz beträchtlich.

Wenn nicht anders erwähnt, sind diese Befehle in */usr/lib/news/bin/maint* zu finden. Denken Sie daran, diese Befehle nur als news auszuführen. Die Ausführung dieser Befehle als Superuser kann dazu führen, daß die Dateien von C News nicht mehr gelesen werden können.

newsdaily

Der Name sagt es bereits: Führen Sie es einmal täglich aus. Es ist ein wichtiges Script, mit dem Sie Ihre Log-Dateien klein halten, wobei jeweils Kopien der letzten drei Läufe aufgehoben werden. Es versucht auch, Anomalien, wie alte Batches in ein- oder ausgehenden Verzeichnissen oder Postings an unbekannte oder moderierte News-Gruppen etc., zu erkennen. Daraus resultierende Fehlermeldungen werden über E-Mail an den Newsmaster geschickt.

newswatch

Dieses Script sollte regelmäßig etwa einmal stündlich ausgeführt werden, um Anomalien im News-System zu entdecken. Es ist für die Erkennung von Problemen gedacht, die einen unmittelbaren Einfluß auf den Betrieb des News-Systems haben. Ein Problembericht wird dann per E-Mail an den Newsmaster geschickt. Überprüft werden Dinge wie alte Lock-Dateien, die nicht entfernt wurden, unbeaufsichtigte Eingabe-Batches und knapper Festplattenspeicher.

addgroup

Fügt Ihrer lokalen Site eine Gruppe hinzu. Der korrekte Aufruf sieht wie folgt aus:

```
addgroup gruppenname y|n|m|=reale_gruppe
```

Das zweite Argument hat dieselbe Bedeutung wie die Option in der *active*-Datei, d. h. jeder kann in diese Gruppe posten(`y`), keiner darf posten (`n`), die Gruppe wird moderiert (`m`), oder daß dies ein Alias für eine andere Gruppe ist (`=reale_gruppe`).

Sie können *addgroup* auch verwenden, wenn die ersten Artikel einer neu angelegten Gruppe früher erscheinen als die `newgroup`-Steuermeldung, die sie anlegen sollte.

delgroup

Erlaubt das lokale Löschen einer Gruppe. Der korrekte Aufruf lautet wie folgt:

```
delgroup gruppenname
```

Die im Spool-Verzeichnis der News-Gruppe verbliebenen Artikel müssen Sie selbst löschen. Alternativ können Sie den Dingen auch ihren natürlichen Lauf lassen (d. h. *expire* abwarten), um sie verschwinden zu lassen.

addmissing

Nimmt fehlende Artikel in die *history*-Datei auf. Führen Sie dieses Script aus, wenn Sie Artikel entdecken, die so aussehen, als würden sie für immer bei Ihnen rumhängen.

newsboot

Dieses Script sollte während der Bootphase des Systems ausgeführt werden. Es entfernt alle Lock-Dateien, die übriggeblieben sind, wenn News-Prozesse bei einem Shutdown unterbrochen wurden. Außerdem schließt es alle Batches und führt noch vorhandene aus, die bei einem Shutdown übriggeblieben sind.

newsrunning

Ist in */usr/lib/news/bin/input* zu finden und wird genutzt, um die Verarbeitung von Batches (z. B. während der normalen Arbeitszeiten) zu unterdrücken. Der Befehl lautet:

```
/usr/lib/news/bin/input/newsrunning off
```

Eingeschaltet wird die Verarbeitung wieder durch Angabe von `on` anstelle des `off`.

Fußnoten

(1)

Es kann einen Unterschied geben zwischen den Gruppen, die auf Ihrer Site vorhanden sind, und den Gruppen, die Ihre Site zu empfangen bereit ist. Beispielsweise könnte die Aboliste comp.all enthalten, was alle News-Gruppen unter der comp-Hierarchie einschließt. Auf Ihrer Site sind aber eine Reihe der comp-Gruppen nicht in Ihrer active-Datei eingetragen. In diese Gruppen gepostete Artikel werden in junk abgelegt.

(2)

Beachten Sie, daß dies die crontab von news sein sollte, damit die Datei-Zugriffsrechte nicht beschädigt werden.

(3)

Es ist nicht unüblich, daß ein in, sagen wir mal, Hamburg geposteter Artikel seinen Weg nach Frankfurt über reston.ans.net in den Niederlanden oder sogar über eine Site in den USA nimmt.

(4)

Wenn Sie die Anzahl der Spool-Verzeichnisse nicht interessiert (weil Sie die einzige Person sind, die diesen Computer benutzt und weil Sie keine megabytegroßen Artikel verfassen), können Sie das Script einfach auf den Eintrag exit 0 reduzieren.

(5)

So wie es mit C News ausgeliefert wird, verwendet compcun compress mit der 12-Bit-Option, weil dies den kleinsten gemeinsamen Nenner für die meisten Sites darstellt. Sie können eine Kopie, beispielsweise compcun16, erzeugen, bei der die 16-Bit-Komprimierung verwendet wird. Die Verbesserung ist allerdings nicht so berauschend.

(6)

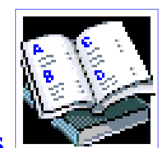
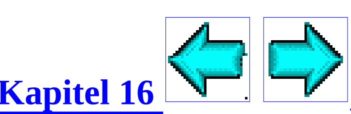
Die Zeit, zu der ein Artikel eingetroffen ist, ist im mittleren Feld der history-Zeile zu finden. Der Wert wird in Sekunden seit dem 1.1.1970 angegeben.

(7)

Ich weiß nicht, warum das passiert, aber bei mir kommt das von Zeit zu Zeit vor.

(8)

Ich würde das auch im Internet nicht versuchen.

Inhaltsverzeichnis  Kapitel 16  Kapitel 18

Kapitel 18
Eine Einführung in NNTP

Aufgrund der völlig verschiedenen Netzwerk-Transportarten vertritt NNTP (verglichen mit C News) einen ganz anderen Ansatz zum Austausch von News. NNTP steht für »Network News Transfer Protocol«. Es ist kein bestimmtes Software-Paket, sondern ein Internet-Standard. Es basiert auf einer Stream-orientierten Verbindung -- üblicherweise über TCP -- zwischen einem Client irgendwo im Netzwerk und einem Server auf einem Host, der die Netnews auf Platte hält. Die Stream-Verbindung erlaubt es dem Client und dem Server, interaktiv den Transfer von Artikeln zu vereinbaren, wobei es nahezu keine Verzögerungen gibt. Auf diese Weise wird die Anzahl doppelt übertragener Artikel klein gehalten. Zusammen mit den hohen Übertragungsraten des Internet ergibt sich so ein News-Transport, der dem jener ursprünglichen UUCP-Netzwerke bei weitem überlegen ist. Während es vor einigen Jahren nicht unüblich war, daß ein Artikel zwei Wochen benötigte, um bis in die letzte Ecke des Usenet zu gelangen, sind es heute weniger als zwei Tage. Im Internet selbst ist es nur noch eine Frage von Minuten.

Verschiedene Befehle erlauben es Clients, Artikel zu empfangen, zu versenden und zu posten. Der Unterschied zwischen dem Versenden und dem Posten liegt darin, daß bei letzterem auch Artikel mit unvollständigen Header-Informationen auftauchen können.(1) Der Abruf von Artikeln kann sowohl über News-Transferclients als auch von Newsreadern erledigt werden. Das macht NNTP zu einem ausgezeichneten Werkzeug, mit dem vielen Clients in einem lokalen Netzwerk der Zugriff auf News gewährt werden kann, ohne daß die beim Einsatz von NFS üblichen Klimmzüge vonnöten wären.

NNTP bietet auch einen aktiven und einen passiven Weg der News-Übertragung, die allgemein als »Pushing« (Schieben) und »Pulling.« (Ziehen) bekannt sind. Pushing ist grundsätzlich dasselbe wie das ihave/sendme-Protokoll von C News. Der Client bietet dem Server einen Artikel über den Befehl »*IHAVE <msgid>*« an, und der Server liefert einen Antwort-Kode zurück, der angibt, daß er den Artikel bereits hat oder daß er ihn haben möchte. Hat er ihn noch nicht, überträgt ihn der Client und schließt den Vorgang durch einen Punkt auf einer separaten Zeile ab.

Dieses Schieben der News hat den einzigen Nachteil, daß es zu einer starken Belastung des Server-Systems führt, weil dieser seine history-Datenbank für jeden einzelnen Artikel durchsuchen muß.

Die andere Technik ist das Ziehen, oder Pulling, von News. Dabei fordert der Client eine Liste aller (verfügbaren) Artikel aus einer Gruppe an, die nach einem bestimmten Datum angekommen sind. Diese Anfrage wird durch den Befehl `NEWNEWS` durchgeführt. Aus der zurückgegebenen Liste von Message-IDs wählt der Client jene aus, die er noch nicht besitzt, wobei auf jeden einzelnen Artikel der `ARTICLE`-Befehl angewendet wird.

Das Problem beim Ziehen von News besteht darin, daß der Server eine strenge Kontrolle darüber ausüben muß, welche Gruppen und Distributionen ein Client anfordern darf. Beispielsweise muß sichergestellt sein, daß kein vertrauliches Material aus lokalen News-Gruppen der Site an nicht autorisierte Clients übertragen wird.

Es existiert eine Reihe weiterer Befehle, mit denen Newsreader den Artikel-Header und die eigentliche Nachricht (den Body) separat lesen. Sogar einzelne Header-Zeilen aus einer Reihe von Artikeln können gelesen werden. Auf diese Weise können alle News auf einem zentralen Host gehalten werden, während alle Benutzer eines (vermutlich lokalen) Netzwerks über NNTP-basierte Clients News lesen und posten können. Dies ist eine Alternative zum in Kapitel 17 beschriebenen Export von News-Verzeichnissen über NFS.



Ein allgemeines Problem von NNTP ist, daß es gut informierten Personen die Möglichkeit bietet, Artikel mit falschen Absenderdaten in den News-Stream einzufügen. Das wird als Fälschen von News (*news faking*) bezeichnet.(2) Eine Erweiterung von NNTP erlaubt die Verwendung von Benutzer-Authentizierungen für bestimmte Befehle.

Es sind eine ganze Reihe von NNTP-Paketen verfügbar. Eines der bekanntesten Pakete ist der NNTP-Daemon, der auch als *Referenz-Implementation* bekannt ist. Er wurde ursprünglich von Stan Barber und Phil Lapsley geschrieben, um die Details von RFC 977 zu verdeutlichen. Die aktuellste Version ist *nntpd-1.5.11*, die nachfolgend besprochen wird. Sie können sich entweder die Quellen besorgen und es selbst kompilieren, oder Sie können *nntpd* aus dem Netkit-Paket nutzen. *nntpd* wird nicht als vorkompilierte Binary angeboten, weil verschiedene Site-spezifische Werte einkompiliert werden müssen.

Das *nntpd*-Paket besteht aus einem Server und zwei Clients, mit denen News geschoben bzw. gezogen werden können. Außerdem gibt es einen Ersatz für den *inews*-Befehl. Sie sind für eine B News-Umgebung konzipiert, mit wenigen Anpassungen aber auch mit C News zu verwenden. Wenn Sie aber mit dem Gedanken spielen, NNTP für mehr einzusetzen, als nur Newsreadern den Zugriff auf Ihren News-Server zu ermöglichen, ist die Referenz-Implementierung nicht wirklich geeignet. Aus diesem Grund behandeln wir an dieser Stelle nur den im *nntpd*-Paket enthaltenen NNTP-Daemon und lassen die Client-Programme außen vor.

Es existiert auch ein Paket namens *InterNet News*, kurz INN, das von Rich Salz geschrieben wurde. News können damit sowohl über NNTP als auch über UUCP transportiert werden, und es ist für größere News-Hubs besser geeignet. Wenn es um den News-Transport über NNTP geht, ist es definitiv besser als *nntpd*. INN ist momentan in der Version *inn-1.4sec* verfügbar. Es existiert ein Kit von Arjan de Vet, mit dem Sie es auf einer Linux-Maschine zum Laufen bringen können. Dieses Kit steht auf **sunsite.unc.edu** im Verzeichnis *system/Mail* zur Verfügung. Wenn Sie INN einrichten wollen, halten Sie sich bitte an die den Quellen beiliegende Dokumentation und an die regelmäßig in *news.software.b* gepostete INN-FAQ.

Installation des NNTP-Servers

Der NNTP-Server heißt *nntpd* und kann, abhängig von der erwarteten Auslastung des News-Systems, auf zwei Arten kompiliert werden. Kompilierte Binärversionen sind nicht verfügbar, weil einige Site-spezifische

Standardwerte fest im ausführbaren Programm integriert sind. Die gesamte Konfiguration wird über in *common /conf.h* definierte Makros erledigt.

nntpd kann als selbständiger Server konfiguriert werden, der während der Boot-Phase aus *rc.inet2* heraus gestartet wird. Alternativ kann er als Dämon konfiguriert werden, der von *inetd* verwaltet wird. Im letzeren Fall müssen Sie die folgende Zeile in Ihre */etc/inetd.conf* aufnehmen:

```
nntp    stream  tcp nowait      news    /usr/etc/in.nntpd    nntpd
```

Wenn Sie *nntpd* als eigenständigen Server konfigurieren, müssen Sie sicherstellen, daß eine solche Zeile in *inetd.conf* auskommentiert ist. In beiden Fällen müssen Sie dafür Sorge tragen, daß die folgende Zeile in */etc/services* auftaucht:

```
nntp    119/tcp   readnews untp    # Network News Transfer Protocol
```

Um alle eingehenden Artikel temporär zwischenspeichern zu können, benötigt *nntpd* auch ein *.tmp*-Verzeichnis in Ihrem News-Spool. Sie können es mit den folgenden Befehlen einrichten:

```
# mkdir /var/spool/news/.tmp
# chown news.news /var/spool/news/.tmp
```

NNTP-Zugriff beschränken

Der Zugriff auf NNTP-Ressourcen wird durch die Datei *nntp_access* im Verzeichnis */usr/lib/news* geregelt. Die Zeilen in dieser Datei beschreiben die fremden Hosts zugestandenen Zugriffsrechte. Jede Zeile verwendet das folgende Format:

```
site   read|xfer|both|no    post|no      [!ausser_diesen_gruppen]
```

Stellt ein Client die Verbindung über den NNTP-Port her, versucht *nntpd* den voll qualifizierten Domain-Namen des Host anhand der IP-Adresse über einen Reverse-Lookup zu ermitteln. Der Host-Name und die IP-Adresse des Client werden in der Reihenfolge ihres Auftretens in der Datei mit dem `site`-Feld verglichen. Vergleiche können exakt oder teilweise übereinstimmen. Bei exakter Übereinstimmung wird der Eintrag verwendet, bei teilweiser Übereinstimmung wird er nur verwendet, wenn nicht noch ein besserer Eintrag folgt. `site` kann in folgender Form angegeben werden:

Host-Name

Der voll qualifizierte Domain-Name eines Host. Stimmt dieser mit dem kanonischen Host-Namen des Client exakt überein, wird der Eintrag verwendet und die nachfolgenden Einträge werden ignoriert.

IP-Adresse

Die IP-Adresse in Dotted Quad Notation. Stimmt sie mit der IP-Adresse des Client überein, wird der Eintrag verwendet und die folgenden Zeilen werden ignoriert.

Domain-Name

Der als *.*domain* angegebene Domain-Name. Enthält der Host-Name des Client den Domain-Namen, wird der Eintrag verwendet.

Netzwerkname

Der Name eines in */etc/networks* spezifizierten Netzwerks. Der Eintrag wird verwendet, wenn die

IP-Adresse des Client mit der dem Netzwerknamen zugewiesenen Netzwerknummer übereinstimmt.

Default

Der String `default` gilt für jeden Client.

Allgemeinere Einträge in der Datei sollten früh in der Datei eingetragen sein, weil alle Übereinstimmungen durch später folgende genauere Übereinstimmungen überschrieben werden.

Das zweite und dritte Feld beschreiben die dem Client zugebilligten Zugriffsrechte. Im zweiten Feld wird bestimmt, ob News gepullt (`read`) oder durch Pushen (`xfer`) übertragen werden dürfen. Wird `both` als Wert verwendet, ist beides möglich; `no` unterdrückt den Zugriff komplett. Im dritten Feld wird festgelegt, ob der Client Artikel posten, d. h. Artikel mit unvollständigen Header-Informationen ausliefern darf, die durch die News-Software vervollständigt werden. Steht im zweiten Feld der Wert `no`, wird das dritte Feld ignoriert.

Das vierte Feld ist optional und enthält eine durch Kommata aufgeteilte Liste von Gruppen, auf die der Client nicht zugreifen darf.

Nachfolgend ein Beispiel für eine *nntp_access*-Datei:

```
#
# Standardmäßig darf jeder News übertragen, aber nicht lesen oder posten.
default                 xfer            no
#
# public.vbrew.com bietet öffentlichen Zugang über Modem.
# Wir erlauben das Lesen und Posten in alle(n) Gruppen außer local.*
public.vbrew.com        read            post    !local
#
# Alle anderen Hosts in der Brauerei dürfen lesen und posten.
*.vbrew.com             read            post
```

NNTP-Autorisierung

Durch Großschreiben der Zugriffs-Tokens wie `xfer` oder `read` in der *nntp_acces*-Datei verlangt *nntpd* für die Durchführung der entsprechenden Operationen eine Autorisierung. Geben Sie beispielsweise `Xfer` oder `XFER` als Zugriffsrecht an, verweigert *nntpd* dem Client so lange die Übertragung von Artikeln zu Ihrer Site, bis die Autorisierung erfolgreich abgeschlossen wurde.

Die Autorisierungs-Prozedur wird über einen neuen NNTP-Befehl namens `AUTHINFO` abgewickelt. Bei diesem Befehl überträgt der Client einen Benutzernamen und ein Paßwort an den NNTP-Server. *nntpd* vergleicht diese mit seiner */etc/passwd*-Datei und prüft gleichzeitig, ob der Benutzer der Gruppe **nntp** angehört.

Die aktuelle Implementierung der NNTP-Autorisierung befindet sich in einem experimentellen Stadium und ist aus diesem Grund nicht besonders portabel gehalten. Daher funktioniert sie nur mit einfachen Paßwort-Datenbanken, Shadow-Paßwörter werden nicht erkannt.

Interaktion von nntpd und C News

Bei der Auslieferung eines Artikels muß *nntpd* ihn an das News-Subsystem weiterleiten. Abhängig davon, ob der Artikel als Ergebnis eines `IHAVE`- oder eines `POST`-Befehls empfangen wurde, wird er an *rnews* oder *inews* übergeben. Statt *rnews* zu starten, können Sie es (während des Kompilierens) so konfigurieren, daß die eingehenden Artikel gesammelt und die daraus resultierenden Batches in */var/spool /news/in.coming* gespeichert werden. Dort werden sie dann von *relaynews* beim nächsten Queue-Lauf benutzt.

Um das ihave/sendme-Protokoll ordnungsgemäß durchführen zu können, muß *nntpd* in der Lage sein, auf die *history*-Datei zugreifen zu können. Sie müssen daher beim Kompilieren darauf achten, daß der Pfad korrekt gesetzt ist. Sie müssen weiterhin sicherstellen, daß C News und *nntpd* sich über das Format Ihrer history-Datei einig sind. C News verwendet für den Zugriff darauf die Hashing-Funktionen von *dbm*. Allerdings gibt es eine Reihe unterschiedlicher und leicht inkompatibler Implementierungen der *dbm*-Bibliothek. Wurde C News mit einer anderen *dbm*-Library gelinkt als der in der Standard-*libc* enthaltenen, muß auch *nntpd* mit dieser Bibliothek gelinkt werden.

Typische Anzeichen dafür, daß *nntpd* und C News im Datenbankformat nicht übereinstimmen, sind Fehlermeldungen im System-Log, wonach *nntpd* sie nicht korrekt öffnen kann. Auch mehrfach über NNTP empfangene Artikel sind ein Indiz. Ein guter Test besteht darin, einen Artikel aus Ihrem Spool-Bereich auszuwählen, Ihren `nntp`-Port über Telnet anzusprechen und den Artikel dann, wie im nachfolgenden Beispiel gezeigt, *nntpd* anzubieten. Natürlich müssen Sie *<msg@id>* durch den Message-ID des Artikels ersetzen, den Sie erneut an *nntpd* übergeben wollen.

```
$ telnet localhost nntp
Trying 127.0.0.1...
Connected to localhost
Escape characters is '^ ]'.
201 vstout NNTP[auth] server version 1.5.11t (16 November 1991) ready at
Sun Feb 6 16:02:32 1194 (no posting)
IHAVE <msg@id>
435 Got it.
QUIT
```

Die Konversation zeigt die korrekte Reaktion von *nntpd*: Die Nachricht »`Got it`« sagt Ihnen, daß der Artikel bereits empfangen wurde. Erhalten Sie statt dessen die Meldung »*335 Ok*«, ist der Lookup in der history-Datei aus irgendeinem Grund fehlgeschlagen. Beenden Sie die Konversation durch Eingabe von Ctrl-D. Was schiefgelaufen ist, können Sie im Systemlog ermitteln. *nntpd* schreibt alle Meldungen in die `daemon`-Einrichtung von *syslog*. Eine inkompatible *dbm*-Bibliothek ist üblicherweise an einer Meldung zu erkennen, die besagt, daß *dbminit* fehlschlug.

Fußnoten

(1)

Wird ein Artikel über NNTP gepostet, fügt der Server mindestens ein Header-Feld, nämlich Nntp-Posting-Host:, hinzu. Es enthält den Host-Namen des Client.

(2)

Dasselbe Problem existiert auch bei SMTP, dem Simple Mail Transfer Protocol.

Kapitel 19
Newsreader-Konfiguration

Ein Newsreader ist ein Programm, mit dem sich Benutzer News-Artikel ansehen, sie speichern und erzeugen. Verschiedene Newsreader wurden auf Linux portiert. Nachfolgend beschreibe ich das grundlegende Setup der drei bekanntesten Newsreader, nämlich *tin*, *trn* und *nn*.

Einer der effektivsten Newsreader ist

```
$ find /var/spool/news -name '[0-9]*' -exec cat {} \; | more
```

Auf diese Weise lesen ganz harte UNIX-Typen ihre News.

Die meisten Newsreader sind aber bei weitem niveauvoller. Sie bieten häufig ein Fullscreen-Interface, bei dem verschiedene Ebenen der vom Benutzer abonnierten Gruppen, eine Übersicht aller Artikel einer Gruppe und individuelle Artikel angeboten werden.

Auf Newsgruppen-Ebene zeigen die meisten Newsreader eine Liste von Artikeln samt Subject-Zeile und Autorennamen. Bei großen Gruppen ist es für den Benutzer schwierig, zusammengehörende Artikel zuzuordnen, obwohl es möglich ist, Antworten auf frühere Artikel zu identifizieren.

Antworten enthalten üblicherweise den ursprünglichen Titel des Artikels, dem der Text »Re:« vorangestellt ist. Zusätzlich kann der Message-ID des Artikels, dessen direkter Nachfolger er ist, in der `References:`-Header-Zeile angegeben werden. Die Sortierung von Artikeln nach diesen beiden Kriterien erzeugt kleine Cluster (eigentlich Bäume) von Artikeln, die *Threads* genannt werden. Eine der Hauptaufgaben beim Schreiben eines Newsreaders ist das Entwerfen eines effizienten Threading-Schemas, weil die dafür verwendete Zeit proportional zum Quadrat der Anzahl der Artikel ist.

An dieser Stelle wollen wir uns nicht damit beschäftigen, wie Benutzerschnittstellen aufgebaut werden. Alle momentan unter Linux verfügbaren Newsreader verfügen über eine gute Hilfefunktion, mit der Sie sich problemlos zurechtfinden sollten.

In den folgenden Abschnitten werden wir uns nur mit administrativen Aufgaben beschäftigen. Die meisten beschäftigen sich mit der Erzeugung von Threads und dem Accounting.

Konfiguration von tin

Unter dem Gesichtspunkt des Threading ist *tin* der mit Abstand vielseitigste Newsreader. Er wurde von Iain Lea geschrieben. Bei der Entwicklung orientierte er sich leicht an einem älteren Newsreader namens *tass* (geschrieben von Rich Skrenta). Das Threading wird erledigt, wenn der Benutzer die Newsgruppe aufruft, und solange Sie nicht über NNTP arbeiten, ist er auch schön schnell.

Auf einem 486DX50 werden für das Threading von 1000 Artikeln ca. 30 Sekunden gebraucht, wenn diese direkt von der Platte gelesen werden. Erfolgt der Zugriff über NNTP auf einen ausgelasteten News-Server, dauert dies etwas über 5 Minuten. Sie können diese Zeit verringern, indem Sie Ihre Indexdatei mit der Option *-u* regelmäßig aktualisieren oder indem Sie *tin* mit der Option *-U* aufrufen.(1)

Normalerweise sichert *tin* seine Threading-Datenbank im Home-Verzeichnis des Benutzers unter *.tin/index*. Das kann eine ziemliche Ressourcen-Verschwendung sein, weshalb Sie nur eine einzelne Kopie an zentraler Stelle speichern sollten. Dies kann erreicht werden, indem Sie *tin* auf Setuid news setzen oder auf einen völlig unprivilegierten Account.(2) *tin* sichert dann alle Thread-Datenbanken unter */var/spool/news/.index*. Für jeden Dateizugriff oder Shell-Aufruf wird die UID dann auf die UID des Benutzers gesetzt, der ihn angefordert hat.(3)

Eine bessere Lösung ist die Installation des *tind*-Indexierungs-Dämons, der die Indexdateien regelmäßig aktualisiert. Dieser Dämon ist allerdings in keiner der Linux-Releases zu finden, d. h. Sie müssen ihn selbst kompilieren. Wenn Sie ein LAN mit einem zentralen News-Server betreiben, können Sie *tind* sogar auf diesem Server laufen und die Clients den Index über NNTP anfordern lassen. Dazu benötigt NNTP eine entsprechende Erweiterung. Patches für *nntpd*, die diese Erweiterung implementieren, sind in den *tin*-Sourcen enthalten.

Manche den Linux-Distributionen beiliegenden *tin*-Versionen bieten noch keine NNTP-Unterstützung an, aber die meisten haben sie bereits integriert. Als *rtin* oder mit der Option *-r* aufgerufen, versucht *tin*, eine Verbindung mit dem in der Datei */etc/nntpserver* oder in der Umgebungsvariablen `NNTPSERVER` eingestellten NNTP-Server aufzubauen. Die Datei *nntpserver* enthält einfach in einer einzigen Zeile den Namen des Servers.

Konfiguration von trn

trn ist ebenfalls ein Nachfolger eines älteren Newsreaders, nämlich *rn* (was für *read news* steht). Das »t« in seinem Namen steht für »threaded«. Er wurde von Wayne Davidson geschrieben.

Im Gegensatz zu *tin* bietet *trn* keine Möglichkeit, die Threading-Datenbank zur Laufzeit zu erzeugen. Statt dessen verwendet es Datenbanken, die mit dem Programm *mthreads* erzeugt wurden, das regelmäßig aus *cron* heraus gestartet werden sollte, um die Indexdateien zu erzeugen.

Lassen Sie *mthreads* nicht laufen, bedeutet das nicht, daß Sie nun keine neuen Artikel lesen können, sondern nur, daß Sie die gesamten »Linux-Ausverkauf durch Novell!!«-Artikel im Artikel-Auswahlmenü verstreut und nicht in einem einzigen Thread vorfinden, den Sie einfach übergehen können.

Um das Threading für bestimmte Newsgruppen zu aktivieren, starten Sie *mthreads* mit einer Liste der

Newsgruppen in der Kommandozeile. Diese Liste ist genauso aufgebaut wie die in der *sys*-Datei:

```
$ mthreads comp,rec,!rec.games.go
```

Dieser Befehl aktiviert das Threading für alle *comp*- und *rec*-Gruppen mit Ausnahme von *rec.games.go* (Go-Spieler brauchen keine Threads). Danach können Sie es einfach ohne Optionen aufrufen, und das Threading wird bei allen neu eingehenden Artikeln durchgeführt. Das Threading aller Gruppen Ihrer *active*-Datei können Sie aktivieren, indem Sie *mthreads* mit der Gruppenliste *all* ausführen.

Wenn Sie News über Nacht empfangen, werden Sie *mthreads* einmal am Morgen ausführen. Wenn nötig, können Sie es aber so häufig ausführen, wie Sie wollen. Sites mit hohem Datenaufkommen sollten *mthreads* im Dämon-Modus ausführen. Wenn Sie es während der Bootphase mit der Option *-d* starten, schiebt es sich selbst in den Hintergrund und prüft alle 10 Minuten, ob neue Artikel eingetroffen sind. Dabei wird automatisch auch das Threading durchgeführt. Um *mthreads* im Dämon-Modus auszuführen, müssen Sie die folgende Zeile in Ihr *rc.news*-Script aufnehmen:

```
/usr/local/bin/rn/mthreads -deav
```

Mit der Option *-a* schaltet *mthreads* automatisch das Threading für neue Gruppen ein, sobald diese erzeugt werden. *-v* aktiviert ausführliche Log-Meldungen, die in die *mthreads*-Logdatei *mt.log* geschrieben werden. Diese ist dort untergebracht, wo auch *trn* installiert ist.

Ältere, nicht mehr verfügbare Artikel müssen regelmäßig aus den Indexdateien entfernt werden. Standardmäßig werden nur Artikel entfernt, deren Nummer unter der kleinsten aktiven Artikelnummer liegt.(4) Artikel, deren Nummer über der kleinsten gültigen Artikelnummer liegt, aber deren Gültigkeitsdauer dennoch abgelaufen ist (weil dem ältesten Artikel über das `Expires:`-Header-Feld eine längere Lebensdauer eingeräumt wurde), können entfernt werden, indem *mthreads* mit der Option *-e* ausgeführt wird. Dies erzwingt einen sog. »erweiterten« expire-Lauf. Läuft *mthreads* im Dämon-Modus, bringt die Option *-e* es in einen solchen erweiterten Modus, der einmal täglich kurz nach Mitternacht durchgeführt wird. Lesen Sie mehr dazu im Kapitel 17.

Konfiguration von nn

nn wurde von Kim F. Storm geschrieben und behauptet, ein Newsreader zu sein, dessen eigentliches Ziel es ist, keine News zu lesen. Sein Name steht für »No News«, und sein Motto lautet »No news is good news. *nn* is better.«

Um dieses ehrgeizige Ziel zu erreichen, wird *nn* mit einer großen Sammlung von Verwaltungs-Tools geliefert. Mit diesen können nicht nur Threads erzeugt, sondern auch die ausführliche Prüfung des Konsistenz dieser Datenbanken durchgeführt werden. Darüber hinaus lassen sich das Accounting verwalten, Benutzerstatistiken erzeugen und Zugriffsrechte vergeben. Es gibt sogar ein Administrations-Programm namens *nnadmin*, mit dem diese ganzen Aufgaben interaktiv durchgeführt werden können. Es ist sehr intuitiv, weshalb wir auch nicht weiter darauf eingehen, sondern uns nur der Generierung der Indexdateien widmen.

Der Thread-Datenbankmanager von *nn* heißt *nnmaster*. Es ist üblich, ihn als Dämon auszuführen, der aus *rc.news* oder *rc.inet2* heraus gestartet wird. Aufgerufen wird er wie folgt:

```
/usr/local/lib/nn/nnmaster -l -r -C
```

Das aktiviert das Threading für alle in Ihrer *active*-Datei vorhandenen Newsgruppen. file.

Ebenso können Sie *nnmaster* regelmäßig über *cron* ausführen, wobei eine Liste der Gruppen übergeben werden muß, die bearbeitet werden sollen. Diese Liste ist der Subskriptions-Liste der *sys*-Datei sehr ähnlich, mit der Ausnahme, daß Leerzeichen anstelle von Kommata verwendet werden. Anstelle des Schlüsselworts `all` muß ein leeres Argument (»«) verwendet werden, wenn alle Gruppen gemeint sind. Der Beispielaufruf lautet:

```
# /usr/local/lib/nn/nnmaster !rec.games.go rec comp
```

Beachten Sie, daß die Reihenfolge wichtig ist. Die Gruppen-Spezifikationen werden von links nach rechts bearbeitet, wobei die erste passende gewinnt. Hätten wir also `!rec.games.go` nach `rec` geschrieben, wäre das Threading für alle Artikel dieser Gruppe durchgeführt worden.

nn bietet verschiedene Möglichkeiten an, ungültige Artikel aus seinen Datenbanken zu entfernen. Die erste besteht darin, die Datenbank zu aktualisieren, indem die Newsgruppen-Verzeichnisse durchsucht und die Einträge entfernt werden, deren Artikel nicht mehr existieren. Das ist die Standardoperation, die durch den Aufruf von *nnmaster* mit der Option *-E* erreicht wird. Das geht einigermaßen zügig, solange Sie dies nicht über NNTP erledigen.

Die zweite Methode verhält sich genau wie der Standard-Aktualisierungslauf von *mthreads*. Dabei werden nur solche Einträge entfernt, deren Artikelnummer unter der gerade gültigen kleinsten Artikelnummer der *active*-Datei liegen. Sie wird durch die Option *-e* gestartet.

Bei der dritten Strategie wird die ganze Datenbank gelöscht, und dann werden alle Artikel erneut eingeordnet. Dieses Verfahren wird verwendet, wenn Sie *nnmaster* mit der Option *-E3* ausführen.

Die Liste der zu aktualisierenden Gruppen wird in derselben Weise wie oben beschrieben über die Option *-F* angegeben. Wenn Sie allerdings *nnmaster* als Dämon betreiben, müssen Sie diesen zuerst anhalten (mit der Option *-k*), bevor der Aktualisierungslauf durchgeführt werden kann. Danach muß er wieder mit seinen normalen Optionen gestartet werden. Die richtige Befehlsfolge für die Aktualisierung aller Gruppen nach der ersten Methode lautet also:

```
# nnmaster -kF ""
# nnmaster -lrC
```

Es gibt noch wesentlich mehr Optionen, mit denen Sie eine Feinabstimmung für das Verhalten von *nn* durchführen können. Wenn Sie schlechte Artikel entfernen oder Artikelsammlungen (Digests) aufsplitten wollen, seien Sie auf die *nnmaster*-Manpage verwiesen.

nnmaster verwendet eine Datei namens *GROUPS*, die in */usr/local/lib/nn* zu finden ist. Wenn sie zu Beginn nicht existiert, wird sie automatisch erzeugt. Für jede Newsgruppe enthält sie eine Zeile, die mit dem Gruppennamen beginnt, dem optional eine Zeitangabe und verschiedene Flags folgen können. Sie können diese Optionen editieren, um das Verhalten der einzelnen Gruppen entsprechend zu beeinflussen, Sie dürfen aber nicht die Reihenfolge verändern, in der die Gruppen aufgeführt sind.(5) Die erlaubten Optionen sowie deren Effekte werden ebenfalls ausführlich in der *nnmaster*-Manpage besprochen.

Fußnoten

(1)

Die Dinge laufen wesentlich schneller, wenn der NNTP-Server das Threading selbst übernimmt und der Client nur die Thread-Datenbank liest. INN-1.4 beispielsweise kann dies.

(2)

Allerdings sollten Sie nicht nobody dafür benutzen. Als Faustregel gilt, daß diesem Benutzer keine Dateien oder Befehle gehören sollten.

(3)

Das ist der Grund für die häßlichen Fehlermeldungen, die Sie erhalten, wenn Sie ihn als Superuser starten. Aber Routinearbeiten sollten Sie sowieso nicht als root erledigen.

(4)

Beachten Sie, daß C News die niedrigste gültige Artikelnummer nicht automatisch setzt. Statt dessen müssen Sie dies mit dem Programm updatemin von Hand erledigen.

(5)

Der Grund dafür liegt darin, daß die Reihenfolge mit den Einträgen in der (binären) MASTER-Datei übereinstimmen muß.

Anhang A
Ein Laplink-Kabel für PLIP

Für ein Laplink-Druckerkabel, das Sie für eine PLIP-Verbindung benutzen, benötigen Sie zwei 25-Pin-Stecker (DB-25) und ein elfadriges Kabel. Das Kabel darf maximal 15 Meter lang sein.

Wenn Sie sich den Stecker genau ansehen, sollten Sie kleine Zahlen unter jedem Pin erkennen, die beim oberen linken Pin mit 1 beginnen (wenn Sie die breitere Seite nach oben halten) und mit 25 beim unteren rechten Pin enden. Für ein Laplink-Kabel müssen Sie die folgenden Pins beider Stecker miteinander verbinden:

```
   D0          2   --   15    ERROR
   D1          3   --   13    SLCT
   D2          4   --   12    PAPOUT
   D3          5   --   10    ACK
   D4          6   --   11    BUSY
   GROUND     25   --   25    GROUND
   ERROR      15   --    2    D0
   SLCT       13   --    3    D1
   PAPOUT     12   --    4    D2
   ACK        10   --    5    D3
   BUSY       11   --    6    D4
```

Alle anderen Pins werden nicht benutzt. Wenn das Kabel abgeschirmt ist, sollte die Schirmung mit dem Metallgehäuse des DB25-Steckers auf nur einer Seite verbunden werden.

Anhang B
Beispiel-Konfigurationsdateien für smail

Dieser Anhang enthält Beispiel-Konfigurationsdateien für eine UUCP-Leafsite (ein Host, der ein LAN über UUCP mit Mail versorgt) in einem lokalen Netzwerk. Die Dateien basieren auf den Beispielen, die in der Source-Distribution von *smail-3.1.28* enthalten sind. Ich habe den kläglichen Versuch unternommen, Ihnen zu erklären, wie diese Dateien arbeiten, Sie sind aber gut beraten, wenn Sie sich die sehr gute *smail(8)*-Manpage ansehen, die diese Dateien sehr ausführlich beschreibt. Haben Sie erst einmal die grundlegende Idee hinter der *smail*-Konfiguration verstanden, lohnt es sich, dieses Dokument zu lesen. Es ist ganz einfach!

Die erste abgedruckte Datei ist die *routers*-Datei, die *smail* eine Reihe von Routern beschreibt. Muß *smail* eine Nachricht an eine gegebene Adresse liefern, wird diese Adresse der Reihe nach an alle Router übergeben, so lange, bis einer paßt. »Passen« bedeutet hier, daß der Router den Zielhost in seiner Datenbank findet, sei es in der *paths*-Datei, in */etc/hosts*, oder welcher Routing-Mechanismus auch immer verwendet wird.

Einträge in *smail*-Konfigurationsdateien beginnen immer mit einem eindeutigen Namen, der den Router, Transport oder Direktor bestimmt. Darauf folgt eine Liste mit Attributen, die das entsprechende Verhalten definieren. Die Liste besteht aus einer Reihe globaler Attribute, wie dem verwendeten *driver*, und privaten Attributen, die nur von dem jeweiligen Driver verstanden werden. Attribute werden durch Kommata, globale und private Attribute hingegen jeweils durch ein Semikolon voneinander getrennt.

Um diese feine Unterscheidung zu verdeutlichen, stellen Sie sich vor, Sie wollen zwei separate *pathalias*-Dateien verwalten; eine enthält die Routing-Information für Ihre Domain, und die zweite enthält globale Routing-Informationen, die möglicherweise aus den UUCP-Maps generiert wurden. Mit *smail* können Sie nun zwei Router in Ihre *routers*-Datei eintragen, die beide den `pathalias`-Driver verwenden. Dieser Treiber durchsucht eine pathalias-Datenbank nach Hostnamen. Es wird erwartet, daß der Dateiname in einem privaten Attribut angegeben wird:

```
#
# pathalias-Datenbank zum Routen innerhalb der Domain
domain_paths:
        driver=pathalias,          # suche Host in Pfaddatei
        transport=uux;             # wenn vorhanden, über UUCP ausliefern
        file=paths/domain,         # Datei ist /usr/lib/smail/paths/domain
        proto=lsearch,             # Datei ist nicht sortiert (lineare Suche)
        optional,                  # ignorieren, wenn Datei nicht existiert
        required=vbrew.com,        # nur nach *.vbrew.com-Hosts suchen
#
# pathalias-Datenbank zum Routen auf Hosts außerhalb der Domain
world_paths:
        driver=pathalias,          # suche Host in Pfaddatei
        transport=uux;             # wenn vorhanden, über UUCP ausliefern
        file=paths/world,          # Datei ist /usr/lib/smail/paths/world
```

```
        proto=bsearch,              # Datei ist mit sort(1) sortiert
        optional,                   # ignorieren, wenn Datei nicht existiert
        -required,                  # keine Domains benötigt
        domain=uucp,                # entferne vor der Suche das ".uucp" am Ende
```

Das zweite globale Attribut, das in beiden oben aufgeführten *routers*-Einträgen verwendet wird, definiert den sog. Transport, d. h. wie die Daten weitergeleitet werden, wenn der Router die passende Adresse findet. In unserem Fall wird die Nachricht mit dem *uux*-Transport ausgeliefert. Transports werden in der Datei *transports* definiert, die noch beschrieben wird.

Sie können genauer bestimmen, mit welchem Transport eine Nachricht ausgeliefert wird, indem Sie eine Methodendatei anstelle des `transports`-Attributs verwenden. Methodendateien ermöglichen die Abbildung von Ziel-Hostnamen auf Transports. Wir gehen an dieser Stelle aber nicht weiter darauf ein.

Die folgende *routers*-Datei definiert Router für ein lokales Netz, die die Resolver-Library abfragen. Allerdings würden Sie auf einem Internet-Host einen Router konfigurieren, der MX-Records verarbeiten kann. Zu diesem Zweck sollten Sie die Kommentarzeichen vor dem alternativen `inet_bind`-Router entfernen, der den eingebauten BIND-Driver von *smail*verwendet.

In einer Umgebung, in der UUCP und TCP/IP gemeinsam genutzt werden, könnten Sie mit dem Problem konfrontiert werden, daß sich Hosts in Ihrer */etc/hosts*-Datei befinden, mit denen Sie nur gelegentlich über SLIP oder PPP Kontakt haben. Normalerweise wollen Sie in solchen Fällen Mail immer noch über UUCP versenden. Um zu verhindern, daß der `inet_hosts`-Driver diese Hosts prüft, müssen Sie sie in die *paths/force*-Datei eintragen. Das ist eine weitere pathalias-ähnliche Datenbank, die durchsucht wird, bevor *smail* den Resolver abfragt.

```
# Eine /usr/lib/smail/routers-Beispieldatei
#
# force -- erzwinge UUCP-Auslieferung an bestimmte Hosts, selbst wenn sie
#       in unserer /etc/hosts stehen
force:
        driver=pathalias,           # suche Host in Pfaddatei
        transport=uux;              # wenn vorhanden, über UUCP ausliefern
        file=paths/force,           # Datei ist /usr/lib/smail/paths/force
        optional,                   # ignorieren, wenn Datei nicht existiert
        proto=lsearch,              # Datei ist nicht sortiert (lineare Suche)
        -required,                  # keine Domains benötigt
        domain=uucp,                # entferne vor der Suche das ".uucp" am Ende
# inet_addrs -- überprüfe Domain-Namen, die literale
#       IP-Adressen wie in janet@[172.16.2.1] enthalten
inet_addrs:
        driver=gethostbyaddr,       # Driver zur Prüfung von IP-Domainliteralen
        transport=smtp;             # Auslieferung mit SMTP über TCP/IP
        fail_if_error,              # Fehler, wenn Adreßformat nicht korrekt
        check_for_local,            # sofort ausliefern, wenn lokaler Host
# inet_hosts -- prüfe Hostnamen mit gethostbyname(3N)
#       kommentieren Sie diese Zeilen aus, wenn Sie die BIND-Version verwenden
        wollen
inet_hosts:
        driver=gethostbyname,       # prüfe Hosts mit Hilfe der Library-Funktion
        transport=smtp;             # SMTP verwenden
        -required,                  # keine Domains benötigt
        -domain,                    # keine Domainendungen definiert
        -only_local_domain,         # nicht auf definierte Domains beschränken
```

```
# inet_hosts -- alternative Version, die BIND verwendet, um auf DNS zuzugreifen
#inet_hosts:
#       driver=bind,              # eingebauten BIND-Driver verwenden
#       transport=smtp;           # mit TCP/IP-SMTP ausliefern
#
#       defnames,                 # normale Domain-Suche verwenden
#       defer_no_connect,         # neuer Versuch, wenn Name-Server down ist
#       -local_mx_okay,           # MX nicht an lokalen Host weiterleiten
#
# pathalias-Datenbank zum Routen innerhalb der Domain
domain_paths:
        driver=pathalias,         # suche Host in Pfaddatei
        transport=uux;            # wenn vorhanden, über UUCP ausliefern
        file=paths/domain,        # Datei ist /usr/lib/smail/paths/domain
        proto=lsearch,            # Datei ist nicht sortiert (lineare Suche)
        optional,                 # ignorieren, wenn Datei nicht existiert
        required=vbrew.com,       # nur nach *.vbrew.com-Hosts suchen
#
# pathalias-Datenbank zum Routen auf Hosts außerhalb der Domain
world_paths:
        driver=pathalias,         # suche Host in Pfaddatei
        transport=uux;            # wenn vorhanden, über UUCP ausliefern
        file=paths/world,         # Datei ist /usr/lib/smail/paths/world
        proto=bsearch,            # Datei ist mit sort(1) sortiert
        optional,                 # ignorieren, wenn Datei nicht existiert
        -required,                # keine Domains benötigt
        domain=uucp,              # entferne vor der Suche das ".uucp" am Ende
# smart_host -- ein teilweise spezifizierter Smart Host-Direktor
#       ist das Attribut smart_path nicht in /usr/lib/smail/config
#       definiert, wird dieser Router ignoriert.
#       das Transport-Attribut wird durch die globale
#       smart_transport-Variable überschrieben
smart_host:
        driver=smarthost,         # Spezial-Driver
        transport=uux;            # Standard-Auslieferung über UUCP
        -path,                    # verwende smart_path Konfigurationsdatei-
                                  # Variable
```

Die Behandlung von Mail für lokale Adressen wird in der Datei *directors* festgelegt. Diese ist genauso aufgebaut wie die *routers*-Datei und enthält eine Liste von Einträgen, die jeweils einen Direktor definieren. Direktors liefern *keine* Nachrichten aus, sondern führen nur alle möglichen Umleitungen über Aliases, Mail-Forwarding und ähnliches durch. like.

Wird Mail an eine lokale Adresse wie beispielsweise *janet* geliefert, gibt *smail* den Benutzernamen an alle aktiven Direktors weiter. Ist ein Direktor erfolgreich, bestimmt er entweder einen Transport, über den die Nachricht ausgeliefert werden soll (z. B. die Mailbox-Datei des Benutzers), oder er generiert eine neue Adresse (beispielsweise nach der Auswertung eines Aliases).

Aus Sicherheitsgründen führen Direktors viele Überprüfungen durch, die ermitteln sollen, ob eine Datei sicher ist oder nicht. Adressen, die auf etwas dubiose Weise gefunden wurden (beispielsweise aus einer allgemein schreibbaren *aliases*-Datei), werden als unsicher markiert. Manche Driver weisen solche Adressen zurück, wie beispielsweise der Transport, der Nachrichten an Dateien liefert.

Darüber hinaus assoziiert *smail* auch einen Benutzer mit jeder Adresse. Alle Schreib-/Leseoperationen werden im Namen dieses Benutzers durchgeführt. Für die Auslieferung in die Mailbox von janet wird die Adresse natürlich auch mit janet assoziiert. Andere Adressen, wie die aus der *aliases*-Datei, assoziieren andere Benutzer, beispielsweise den Benutzer nobody.

Die Details dieser Features entnehmen Sie bitte der *smail(8)*-Manpage.

```
# Eine /usr/lib/smail/directors-Beispieldatei
# aliasinclude -- löse aus Alias-Dateien erzeugte ":include:filename"-Adressen
# auf
aliasinclude:
        driver=aliasinclude,    # verwende Spezial-Driver
        nobody;                 # greife als Benutzer "nobody" auf Datei zu,
                                # wenn unsicher
        copysecure,             # übernehme Rechte von Alias-Direktor
        copyowners,             # übernehme Besitzer von Alias-Direktor
# forwardinclude -- löse aus Forward-Dateien erzeugte
#       ":include:filename"-Adressen auf
forwardinclude:
        driver=forwardinclude,    # verwende Spezial-Driver
        nobody;                   # greife als Benutzer "nobody" auf Datei zu,
                                  # wenn unsicher
        checkpath,                # prüfe Zugriff auf Pfad
        copysecure,               # übernehme Rechte vom Forwarding-Direktor
        copyowners,               # übernehme Besitzer vom Forwarding-Direktor
# aliases -- suche nach in der Datenbank gespeicherten Alias-Auflösungen
aliases:
        driver=aliasfile,         # Allzweck-Aliasing-Direktor
        -nobody,                  # alle Adressen sind per Voreinstellung
                                  # bereits mit nobody assoziiert
        sender_okay,              # Sender aus Erweiterungen nicht entfernen
        owner=owner-$user;        # Probleme an Benutzeradresse melden
        file=/usr/lib/aliases,    # Voreinstellung: sendmail-Kompatibilität
        modemask=002,             # sollte nicht allgemein geschrieben werden
                                  # können
        optional,                 # ignorieren, wenn Datei nicht existiert
        proto=lsearch,            # unsortierte ASCII-Datei
# dotforward -- löse .forward-Dateien im Home-Verzeichnis des Benutzers auf
dotforward:
        driver=forwardfile,       # Allzweck-Forwarding-Direktor
        owner=real-$user,         # Probleme an Benutzer-Mailbox melden
        nobody,                   # Benutzer ist nobody, wenn unsicher
        sender_okay;              # Sender niemals aus Erweiterungen entfernen
        file=~/.forward,          # .forward-Datei im Home-Verzeichnis
        checkowner,               # Benutzer kann diese Datei besitzen
        owners=root,              # aber auch root kann diese Datei besitzen
        modemask=002,             # sollte nicht allgemein geschrieben werden
                                  # können
        caution=0-10:uucp:daemon, # nicht als root oder Daemon
        # besondere Vorsicht bei Home-Verzeichnissen auf entfernten Rechnern
        unsecure="~ftp:~uucp:~nuucp:/tmp:/usr/tmp",
# forwardto -- löse "Forward to "-Zeile zu Beginn
```

```
#       der Mailbox-Datei eines Benutzers auf
forwardto:
        driver=forwardfile,
        owner=Postmaster,          # Fehler an Postmaster melden
        nobody,                    # Benutzer ist nobody, wenn unsicher
        sender_okay;               # Sender aus Erweiterungen nicht entfernen
        file=/var/spool/mail/${lc:user}, # Position der Benutzer-Mailbox
        forwardto,                 # aktiviere "Forward to "-Prüfung
        checkowner,                # Benutzer kann diese Datei besitzen
        owners=root,               # aber auch root kann diese Datei besitzen
        modemask=0002,             # Unter System V kann Group-Mail schreiben
        caution=0-10:uucp:daemon, # nicht als root oder Daemon
# user -- überprüfe Benutzer auf dem lokalen Host mit Lieferung zu deren
#       Mailboxen
user:   driver=user;               # Driver zur Prüfung der Benutzernamen
        transport=local,           # lokaler Transport erfolgt in Mailboxen
# real_user -- überprüfe Benutzernamen, wenn der Prefix mit dem String
#       "real-" beginnt
real_user:
        driver=user;               # Driver zur Prüfung der Benutzernamen
        transport=local,           # lokaler Transport erfolgt in Mailboxen
        prefix="real-",            # Prüfe z. B. "real-root"
# lists -- löse Mailing-Listen auf, die unter /usr/lib/smail/lists
#       gespeichert sind
lists:  driver=forwardfile,
        caution,                   # alle Adressen sind mit Vorsicht zu genießen
        nobody,                    # und mit dem Benutzer nobody assoziiert
        sender_okay,               # Sender NICHT entfernen
        owner=owner-$user;         # Besitzer der Liste
        # wandele den Namen der Mailing-Liste in Kleinbuchstaben
        file=lists/${lc:user},
```

Wurde eine Nachricht erfolgreich geroutet oder geleitet, übergibt *smail* die Nachricht an den vom Router oder Direktor spezifizierten Transport, der die Adresse prüft. Diese Transports sind in der Datei *transports* definiert. Auch ein Transport wird durch eine Reihe globaler und privater Optionen definiert.

Die wichtigste Option, die in jedem Eintrag definiert wird, ist der Driver, der den Transport übernimmt. Dazu gehört beispielsweise der Driver `pipe`, der den im `cmd`-Attribut angegebenen Befehl ausführt. Darüber hinaus gibt es eine ganze Reihe globaler Attribute, die vom Transport verwendet werden können, um den Nachrichten-Header und möglicherweise den -Body umzuwandeln. Beispielsweise läßt das Attribut `return_path` den Transport ein `Return-Path`-Feld in den Header einfügen. Durch das Attribut `unix_from_hack` wird jeder Zeile, die mit dem Wort `From` beginnt, ein >-Zeichen vorangestellt.

```
# Eine /usr/lib/smail/transports-Beispieldatei
# local -- Mail an lokale Benutzer ausliefern
local:  driver=appendfile,         # Nachricht an Datei anhängen
        return_path,               # Return-Path:-Feld einfügen
        from,                      # From_-Zeile verwenden
        unix_from_hack,            # > vor From im Body einfügen
        local;                     # lokale Forms verwenden
        file=/var/spool/mail/${lc:user}, # Position der Mailbox-Dateien
        group=mail,                # Besitzergruppe für System V
```

```
        mode=0660,                # Gruppe, auf die die Mail zugreifen kann
        suffix="\n",              # Extra-Zeilenvorschub anhängen
# pipe -- Mail an Shell-Befehle weiterleiten
pipe:   driver=pipe,              # Nachricht an anderes Programm pipen
        return_path,              # Return-Path:-Feld einfügen
        from,                     # From_ -Zeile verwenden
        unix_from_hack,           # > vor From im Body einfügen
        local;                    # lokale Forms verwenden
        cmd="/bin/sh -c $user", # Adresse an Bourne Shell
        parent_env,               # Umgebungsinfo von Parent-Adresse
        pipe_as_user,             # mit Benutzer-ID assoziierte Adresse
                                  # verwenden
        ignore_status,            # Exit-Status ungleich 0 ignorieren
        ignore_write_errors,      # Schreibfehler ignorieren (gebrochene Pipe)
        umask=0022,               # umask für Child-Prozeß
        -log_output,              # stdout/stderr nicht loggen
# file -- Mail in Dateien liefern
file:   driver=appendfile,
        return_path,              # Return-Path:-Feld einfügen
        from,                     # From_-Zeile verwenden
        unix_from_hack,           # > vor From im Body einfügen
        local;                    # lokale Forms für Lieferung verwenden
        file=$user,               # Datei wird aus Adresse genommen
        append_as_user,           # verwende mit Benutzer-ID assoziierte Adresse
        expand_user,              # löse ~ und $ innerhalb der Adresse auf
        suffix="\n",              # Extra-Zeilenvorschub anhängen
        mode=0600,                # Zugriffsrechte auf 600 setzen
# uux -- an rmail-Programm an einer entfernten UUCP-Site weiterleiten
uux:    driver=pipe,
        uucp,                     # UUCP-Adreßformate verwenden
        from,                     # From_-Zeile verwenden
        max_addrs=5,              # maximal 5 Adressen pro Aufruf
        max_chars=200;            # maximal 200 Zeichen für Adressen
        cmd="/usr/bin/uux -- -r -a$sender -g$grade $host!rmail $(($user)$)",
        pipe_as_sender,           # uucp-Logs enthalten Rufer
        log_output,               # Fehlermeldungen für gebouncte Nachrichten
                                  # speichern
#       defer_child_errors,       # erneuter Versuch, wenn uux Fehler meldet
# demand -- Lieferung an unmittelbar pollendes entferntes rmail-Programm
demand: driver=pipe,
        uucp,                     # UUCP-Adreßformate verwenden
        from,                     # From_-Zeile verwenden
        max_addrs=5,              # maximal 5 Adressen pro Aufruf
        max_chars=200;            # maximal 200 Zeichen für Adressen
        cmd="/usr/bin/uux -- -a$sender -g$grade $host!rmail $(($user)$)",
        pipe_as_sender,           # uucp-Logs enthalten Rufer
        log_output,               # Fehlermeldungen für gebouncte Nachrichten
                                  # speichern
#       defer_child_errors,       # erneuter Versuch, wenn uux Fehler meldet
# hbsmtp -- halbgares BSMTP. Die Ausgabedateien müssen
#       regelmäßig bearbeitet und über UUCP verschickt werden.
hbsmtp: driver=appendfile,
```

```
        inet,                       # verwende RFC 822-Adressierung
        hbsmtp,                     # "batched SMTP" ohne HELO und QUIT
        -max_addrs, -max_chars;     # kein Limit bei Zahl der Adressen
        file="/var/spool/smail/hbsmtp/$host",
        user=root,                  # Datei gehört root
        mode=0600,                  # kann nur von root gelesen und geschrieben
                                    # werden
# smtp -- Lieferung mit SMTP über TCP/IP
smtp:   driver=tcpsmtp,
        inet,
        -max_addrs, -max_chars;     # kein Limit bei Zahl der Adressen
        short_timeout=5m,                 # Timeout für kurze Operationen
        long_timeout=2h,                  # Timeout für längere SMTP-Operationen
        service=smtp,                     # Verbindung mit dem Service-Port
# bei Internet: die folgenden vier Zeilen auskommentieren
#       use_bind,                         # löse MX- und mehrfache A-Records auf
#       defnames,                         # verwende voreingestellte Domainsuche
#       defer_no_connect,                 # erneuter Versuch, wenn Name-Server
                                          # nicht hochgefahren
#       -local_mx_okay,                   # MX nicht an lokalen Host weiterleiten
```

Anhang C
Copyright und Lizenzinformationen

In diesem Anhang haben wir die »GNU General Public License« (kurz *GPL* oder *Copyleft* genannt) im englischen Original abgedruckt. Linux wird zu den Bedingungen der GPL lizensiert. Wir haben die GPL hier aufgenommen, um etwas Klarheit zum Copyright-Status von Linux zu schaffen. Linux ist *keine* Shareware und ist auch *nicht* Teil der Public Domain. Ein Großteil des Linux-Kernels ist Copyright \xa9 1993 Linus Torvalds. Andere Software sowie Teile des Kernels sind von den jeweiligen Autoren urheberrechtlich geschützt. Das bedeutet also, daß Linux urheberrechtlich geschützt *ist*. Trotzdem darf es zu den Bedingungen der GPL weitergegeben werden.

Viele Netzwerk-Utilities und -Daemons wurden von der Berkeley-UNIX-Software-Distribution auf Linux portiert. Die Copyright-Vereinbarung, zu der die University of Berkeley diese Software der Öffentlichkeit zugänglich macht, unterscheidet sich von der GPL, die den Kernel und die Free Software Foundation betrifft. Dieser Anhang enthält das Quellkode-Copyright und Distributionsinformationen, wie sie in BSD-Quelldateien erscheinen.

Zu Beginn finden Sie vollständige Copyright-Informationen zu diesem Buch. Dabei handelt es sich um die Fortführung der am Anfang dieses Buchs stehenden Copyright-Hinweise.

Linux Network Administrator's Guide -- Copyright Information

Copyright © 1993-1995 Olaf Kirch
Kattreinstr. 38, 64295 Darmstadt, Germany
okir@monad.swb.de

UNIX is a trademark of X/Open Inc.
Linux is not a trademark, and has no connection to UNIX or X/Open Inc.

The version of the *Linux Network Administrator's Guide* that has been distributed through the Linux Documentation Project may be reproduced under the conditions described here.

The O'Reilly & Associates, Inc. edition of this book cannot be photocopied or otherwise reproduced.

The Linux Network Administrator's Guide may be reproduced and distributed in whole or in part, subject to the following conditions:

1. The copyright notice above and this permission notice must be preserved complete on all complete or partial copies.
2. Any translation or derivative work of *The Linux Network Administrator's Guide* must be approved by the author in writing before distribution.
3. If you distribute *The Linux Network Administrator's Guide* in part, instructions for obtaining the complete version of *The Linux Network Administrator's Guide* must be included, and a means for obtaining a complete version provided.
4. Small portions may be reproduced as illustrations for reviews or quotes in other works without this permission notice if proper citation is given.
5. The GNU General Public License referenced below may be reproduced under the conditions given within it.
6. Several sections of this document are held under separate copyright. When these sections are covered by a different copyright, the seperate copyright is noted. If you distribute The Linux Network Administrator's Guide in part, and that part is, in whole, covered under a seperate, noted copyright, the conditions of that copyright apply.

Exceptions to these rules may be granted for academic purposes: Write to Olaf Kirch at the above address, or email *okir@monad.swb.de,* and ask. These restrictions are here to protect us as authors, not to restrict you as educators and learners.

All source code in *The Linux Network Administrator's Guide* is placed under the GNU General Public License. This appendix contains a copy of the GNU »GPL.«

The author is not liable for any damages, direct or indirect, resulting from the use of information provided in this document.

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright \xa9 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software -- to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions for Copying, Distribution, and Modification

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The »Program«, below, refers to any such program or work, and a »work based on the Program« means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term »modification«.) Each licensee is addressed as »you«.

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.
2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   1. You must cause the modified files to carry prominent notices stating that you changed the files

and the date of any change.

2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   1. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   2. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   3. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the

limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and »any later version«, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.
12. NO WARRANTY

    BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM »AS IS« WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

    END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the »copyright« line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright \xa9 19yy <name of

author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands »show w« and »show c« should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than »show w« and »show c«; they could even be mouse-clicks or menu items -- whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a »copyright disclaimer« for the program, if necessary. Here is a sample; alter the names: Yoyodyne, Inc., hereby disclaims all copyright interest in the program &guilsinglright;Gnomovision&guilsinglleft; (which makes passes at compilers) written by James Hacker.

<*signature of Ty Coon*>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

The Berkeley Software Distribution Copyright

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product includes software developed by the University of California, Berkeley and its contributors.

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS »AS IS« AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Anhang D
SAGE: Die Gilde der Systemadministratoren

Wenn Sie nicht alle gewünschten Informationen durch das Lesen der Dokumentation bzw. durch Postings in *comp.os.linux.** erhalten, sollten Sie vielleicht der »System Administrators Guild« (SAGE) beitreten, die von Usenix unterstützt wird. Das Hauptziel von SAGE ist die Etablierung der Systemadministration als ernsthaften Beruf. SAGE bringt System- und Netzwerk-Administratoren zusammen, um die berufliche und technische Entwicklung zu fördern, Probleme und deren Lösungen zu verbreiten. Auch die Diskussion zwischen Benutzern, Managern und Anbietern zu Themen der Systemadministration kommt nicht zu kurz.

Nachfolgend einige aktuelle SAGE-Initiativen:

- Gemeinsam mit USENIX Sponsoring der jährlich stattfindenden, sehr erfolgreichen Systemadministrations-Konferenzen (LISA).
- Veröffentlichung der von Tina Darmaohray herausgegebenen »Job Descriptions for System Administrators,«. Dies ist der erste in einer Reihe praktischer Bücher und Ressourcen-Führer, die sich mit Themen und Techniken der Systemadministration beschäftigen.
- Aufbau einer Archiv-Site (**ftp.sage.usenix.org**) mit Dokumenten von Systemadministrations-Konferenzen sowie sysadmin-nahen Dokumenten.
- Aufbau von Arbeitsgruppen zu für Systemadministratoren wichtigen Bereichen wie Jobs, Veröffentlichungen, Verordnungen, elektronische Informationsverteilung, Aus- und Weiterbildung, Anbietern und Standards.

Wenn Sie mehr über die USENIX-Association und SAGE erfahren wollen, wenden Sie sich an das Büro der USENIX-Association in den USA, Telefon 001 510 528 8649, oder per E-Mail an *office@usenix.org*. Elektronische Informationen können Sie über *info@usenix.org* beziehen. Der jährliche Mitgliedsbeitrag für SAGE liegt bei nur $25 (Sie müssen gleichzeitig USENIX-Mitglied sein). Im Beitrag enthalten sind ein Abo von *;login:* und Computing Systems, eine vierteljährlich erscheinende technische Referenz, und vergünstigte Konditionen für Konferenzen und Symposien, Preisnachlässe beim Kauf von Veröffentlichungen und andere Dienste.

Glossar

Ein Problem bei Netzwerken ist, die Bedeutung all der Abkürzungen und Fachausdrücke zu behalten. Hier ist eine Liste mit in diesem Buch häufig verwendeten Abkürzungen und Fachausdrücken, versehen jeweils mit einer kurzen Erläuterung.

ACU

Automatic Call Unit. Ein Modem.

ARP

Address Resolution Protocol. Wird verwendet, um IP-Adressen in Ethernet-Adressen umzuwandeln.

ARPA

Advanced Research Project Agency, später DARPA. Gründer des Internet.

ARPANET

Der Vorläufer des heutigen Internet. Ein experimentelles, von der amerikanischen U.S. Defense Advanced Research Project Agency (DARPA) gegründetes Netzwerk.

Assigned Numbers

Der Titel eines regelmäßig veröffentlichten RFC, das alle öffentlich zugewiesenen Nummern aufführt, die für die unterschiedlichsten Zwecke bei TCP/IP-Netzwerken verwendet werden. Beispielsweise ist eine Liste aller Port-Nummern bekannter Dienste wie *rlogin*, *telnet* etc. enthalten. Die neueste Ausgabe dieses Dokuments ist RFC 1340.

Bang Path

Bei UUCP-Netzwerken eine spezielle Notation für den Pfad von einer UUCP-Site zu einer anderen. Der Name stammt von den Ausrufezeichen (»bangs«) ab, die verwendet werden, um die verschiedenen Hostnamen zu trennen. Beispiel: *foo!bar!ernie!bert* bezeichnet einen Pfad zum Host **bert**, der (in dieser Reihenfolge) über **foo**, **bar** und **ernie** führt.

BBS

Bulletin Board System. Ein Dialup-Mailbox-System.

BGP

Border Gateway Protocol. Ein Protokoll zum Austausch von Routing-Informationen zwischen autonomen Systemen.

BIND

Der Berkeley Internet Name Domain-Server. Die Implementierung eines DNS-Servers.

BNU

Basic Networking Utilities. Die momentan gängigste UUCP-Variante. Auch als »HoneyDanBer UUCP« bekannt. Dieser Name ist von den Namen der Autoren (P. Honeyman, D.A. Novitz und B.E. Redman) abgeleitet.

Broadcast-Netzwerk

Ein Netzwerk, das es einer Station erlaubt, Datagramme an alle anderen Stationen im Netzwerk gleichzeitig zu adressieren.

BSD

Berkeley Software Distribution. Eine UNIX-Variante.

kanonischer Hostname

Der primäre Name eines Hosts innerhalb des Domain Name System. Dies ist der einzige Name des Host, an den auch ein A-Record gebunden ist, und der zurückgegeben wird, wenn ein Reverse Lookup durchgeführt wird.

CCITT

Comiteé Consultatif International de Télégraphique et Téléphonique. Eine internationale Organisation von Telefondiensten etc.

CSLIP

Compressed Serial Line IP. Ein Protokoll zur Übertragung von IP-Paketen über serielle Leitungen, das die Header der meisten TCP/IP-Datagramme komprimiert.

DNS

Domain Name System. Eine verteilte Datenbank, die im Internet verwendet wird, um Hostnamen in IP-Adressen zu wandeln.

EGP

External Gateway Protocol. Ein Protokoll zum Austauschen von Routing-Informationen zwischen autonomen Systemen.

Ethernet

Umgangssprachlich der Name für eine bestimmte Netzwerk-Ausrüstung. Technisch meint Ethernet den Teil einer Reihe von Standards, die vom IEEE festgelegt wurden. Die Ethernet-Hardware verwendet ein einziges Stück Kabel, häufig Koax-Kabel, um eine Reihe von Hosts zu verbinden. Sie ermöglicht Übertragungsraten von bis zu 10 Mbps. Das Ethernet-Protokoll definiert die Art und Weise, in der Hosts über dieses Kabel kommunizieren.(1)

FQDN

Fully Qualified Domain Name. Ein Hostname mit anhängigem Domainnamen, und auf diese Weise ein gültiger Index auf die Domain Name-Datenbank.

FTP

File Transfer Protocol. Das Protokoll, auf dem einer der bekanntesten Datei-Transfer-Dienste basiert, und nach dem er auch benannt ist.

FYI

»For Your Information.« Eine Reihe von Dokumenten mit informellen Informationen zu Internet-Themen.

GMU

Groucho-Marx-Universität. Eine fiktive Universität, die als Beispiel in diesem Buch verwendet wird.

GNU

»GNU's not Unix« -- Dieses rekursive Akronym ist der Name eines Projekts der Free Software Association. Ziel ist die Bereitstellung einer Reihe von UNIX-Tools, die zueinander passen und kostenlos kopiert und benutzt werden dürfen. GNU-Software darf unter den Bedingungen der sog. »GNU General Public License« (GPL), auch »Copyleft« genannt, verwendet werden. Die GPL ist im Kapitel Anhang C, *Copyright und Lizenzinformationen* abgedruckt.

HoneyDanBer

Der Name einer UUCP-Variante. Siehe auch BNU.

Host

Ganz allgemein ein Netzwerk-Knoten: Etwas, was in der Lage ist, Netzwerk-Nachrichten zu empfangen und zu versenden. Das ist üblicherweise ein Computer, aber auch X-Terminals oder intelligente Drucker sind denkbar.

ICMP

Internet Control Message Protocol. Ein Netzwerk-Protokoll, das von IP verwendet wird, um Informationen über Fehler an den sendenden Host zurückzugeben etc.

IEEE

Institute of Electrical and Electronics Engineers. Noch eine Standardisierungs-Organisation. Aus Sicht des UNIX-Benutzers ist deren vielleicht wichtigste Errungenschaft der POSIX-Standard. Dieser definiert verschiedene Aspekte von UNIX-Systemen, angefangen von Interfaces für Systemaufrufe über die Semantik bis hin zu Werkzeugen für die Administration. Daneben hat das IEEE auch die Spezifikationen für Ethernet, Token-Ring- und Token-Bus-Netzwerke entwickelt. Ein weitverbreiteter Standard für die binäre Repräsentation realer Zahlen stammt ebenfalls vom IEEE.

IETF

Internet Engineering Task Force.

Internet

Ein Computernetzwerk, das sich aus einer Reihe individueller, kleinerer Netzwerke zusammensetzt.

Das Internet

Ein bestimmtes, weltweites Internet.

IP

Internet Protocol. Ein Netzwerk-Protokoll.

ISO

International Standards Organization.

ISDN

Integrated Services Digital Network. Relativ neue Telekommunikations-Technologie, die digitale anstelle analoger Techniken benutzt.

LAN

Local Area Network. Ein kleines Computernetzwerk.

MX

Mail Exchanger. Ein DNS-Ressource-Record-Typ, der verwendet wird, um einen Host als Mail-Gateway für eine Domain zu kennzeichnen.

Netzwerk, paketorientiert

Ein Netzwerk, bei dem Daten unmittelbar weitergeleitet werden, indem sie in kleine Pakete zerlegt werden, die dann individuell übertragen werden. Paketorientierte Netzwerke benötigen semipermanente oder permanente Verbindungen.

Netzwerk, »Store-and-Forward«

Das genaue Gegenteil eines paketorientierten Netzwerks. Das Netzwerk überträgt die Daten »an einem Stück« und arbeitet nicht mit permanenten Verbindungen. Statt dessen wird die Verbindung zwischen Hosts nur in regelmäßigen Intervallen aufgebaut, und alle Daten werden auf einmal übertragen. Das bedeutet, daß die Daten zwischengespeichert werden müssen, bis die Verbindung aufgebaut wird.

NFS

Network File System. Ein Standard Netzwerk-Protokoll und eine Software-Sammlung, die den transparenten Zugriff auf entfernte Festplatten (»remote disks«) ermöglicht.

NIS

Network Information System. Eine RPC-basierte Anwendung, die es erlaubt, Konfigurationsdateien wie beispielsweise die Paßwortdatei auf verschiedenen Hosts gemeinsam zu nutzen. Siehe auch YP.

NNTP

Network News Transfer Protocol. Ein Protokoll zur Übertragung von News über TCP-Netzwerk-Verbindungen.

Oktett

Im Internet steht dieser technische Ausdruck für 8 Bit-Werte. Er wird häufiger verwendet als *Byte*, weil es im Internet Maschinen gibt, die mit anderen Byte-Größen arbeiten.

OSI

Open Systems Interconnection. Ein ISO-Standard für Netzwerk-Software.

Pfad

Wird in UUCP-Netzwerken häufig als Synonym für *Route* verwendet. Siehe auch *Bang Path*.

PLIP

Parallel Line IP. Ein Protokoll zum Austausch von IP-Paketen über parallele Leitungen (z. B. einen Druckerport).

Port, TCP oder UDP

Für TCP und UDP sind Ports die Endpunkte für einen Dienst. Bevor ein Prozeß einen Dienst anbieten oder darauf zugreifen kann, muß er zuerst einen Port beanspruchen (binden). Zusammen mit den IP-Adressen der Hosts identifizieren Ports eindeutig die zwei Punkte einer TCP-Verbindung.

Portmapper

Der Portmapper ist der Vermittler zwischen den Programmnummern, die von RPC als Identifikation für individuelle RPC-Server verwendet werden, und den TCP- und UDP-Portnummern, an denen diese Dienste horchen.

PPP

Point-to-Point Protocol. PPP ist ein flexibles und schnelles Protokoll der Verbindungsschicht. Es wird verwendet, um verschiedene Netzwerk-Protokolle wie IP oder IPX über eine Punkt-zu-Punkt-Verbindung zu senden. Neben der Verwendung bei seriellen Links (Modem) kann PPP auch als Link-Level-Protokoll auf ISDN aufgesetzt werden.

RARP

Reverse Address Resolution Protocol. Ermöglicht es Hosts, ihre IP-Adresse während der Boot-Phase zu ermitteln.

Resolver

Eine Bibliothek (Library), die Hostnamen auf IP-Adressen abbildet und umgekehrt.

Ressource Record

Die grundlegende Informationseinheit in einer DNS-Datenbank, üblicherweise mit »RR« abgekürzt. Jeder Record gehört einem bestimmten Typ und einer bestimmten Klasse an. Beispielsweise ist ein Record, der Hostnamen auf IP-Adressen abbildet, vom Typ A (für »Address«) und gehört der Klasse IN (für »Internet Protocol«) an.

Reverse Lookup

Die Ermittlung eines Hostnamens anhand einer gegebenen IP-Adresse. Bei DNS wird dies durch einen Lookup nach der IP-Adresse des Host in der in-addr.arpa-Domain erreicht.

RFC

Request For Comments. Eine Reihe von Dokumenten, die die Internet-Standards beschreiben.

RIP

Routing Information Protocol, ein Routing-Protokoll. Wird verwendet, um innerhalb eines (kleinen) Netzwerks dynamisch Routen einzustellen.

Route

Die Reihe von Hostrechnern, die ein Datenpaket durchlaufen muß, um vom Ursprungs- zum Zielhost zu gelangen. Eine entsprechende Route zu finden, wird auch als *Routing* bezeichnet.

Routing-Dämon

In einem größeren Netzwerk sind Änderungen in der Netzwerk-Topologie nur schwer von Hand durchzuführen. Aus diesem Grund werden Einrichtungen verwendet, die die aktuellen Routing-Informationen an die zugehörigen Hosts weiterleiten. Diese Technik wird dynamisches Routing genannt; Routing-Informationen werden über *Routing-Dämonen* ausgetauscht, die auf zentralen Hosts im Netzwerk laufen. Die eingesetzten Protokolle werden *Routing-Protokolle* genannt.

RPC

Remote Procedure Call. Ein Protokoll, mit dem Sie Prozeduren innerhalb eines Prozesses auf einem entfernten Host ausführen können.

RR

Abkürzung für *Resource Record*.

RS-232

Ein sehr weit verbreiteter Standard für serielle Schnittstellen.

RTS/CTS

Der umgangssprachliche Name für den Hardware-Handshake, der von zwei über RS-232 kommunizierenden Einrichtungen durchgeführt wird. Der Name ist von den beiden dafür zuständigen Leitungen RTS (»Ready To Send«) und CTS (»Clear To Send«) abgeleitet.

RTM Internet Worm

Ein virusähnliches Programm, das verschiedene Mängel von VMS und BSD-UNIX 4.3 nutzte, um sich im Internet auszubreiten. Diverse »Versehen« im Programm haben dafür gesorgt, daß es sich unkontrolliert vervielfältigt und damit große Teile des Internet lahmgelegt hat. RTM sind die Initialen des Autors (Robert T. Morris), die er im Programm belassen hat.

Site

Eine Ansammlung von Host-Rechnern, die sich nach außen wie ein einziger Netzwerk-Knoten verhalten. Aus der Sicht des Internet würden Sie beispielsweise die Groucho-Marx-Universität als Site bezeichnen, unabhängig davon, wie komplex deren internes Netzwerk wirklich ist.

SLIP

Serial Line IP. Ein Protokoll, mit dem Sie IP-Pakete über serielle Leitungen übertragen können. Siehe auch *CSLIP*.

SMTP

Simple Mail Transfer Protocol. Wird für die Übertragung von Mail über TCP-Verbindungen eingesetzt, kann aber auch genutzt werden, um Mail über UUCP-Links (»batched SMTP«) zu transportieren.

SOA

Start of Authority. Ein DNS Resource Record-Typ.

System V

Eine UNIX-Variante.

TCP

Transmission Control Protocol. Ein Netzwerk-Protokoll.

TCP/IP

Lässige Beschreibung für die Internet-Protokolle als Ganzes.

UDP

User Datagram Protocol. Ein Netzwerk-Protokoll.

UUCP

Unix to Unix Copy. Eine Reihe von Netzwerk-Transport-Befehlen für Dialup-Netzwerke.

Version 2 UUCP

Eine alternde UUCP-Variante.

Virtuelles Bier

Das Lieblingsgetränk jedes Linuxers. Die erste Erwähnung von virtuellem Bier, an die ich mich erinnern kann, stand in den Release-Notes zum Linux 0.98.X-Kernel, wo sich Linus bei den »Oxford Beer Trolls« für eine Lieferung virtuellen Bieres bedankte.

»Well-Known Services« (Bekannte Dienste)

Dieser Ausdruck wird häufig für gängige Netzwerkdienste wie *telnet* und *rlogin* verwendet. Aus technischer Sicht beschreibt er alle Dienste, denen offiziell eine Portnummer im »Assigned Numbers RFC« zugewiesen wurde.

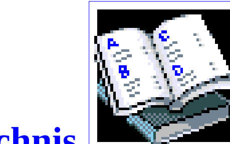
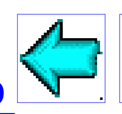
YP

Yellow Pages. Ein älterer Name für NIS, der aber nicht länger verwendet wird, weil »Yellow Pages« (gelbe Seiten) ein Warenzeichen der British Telecom ist. Dessen ungeachtet haben die meisten NIS-Utilities ihre Namen beibehalten, die mit *yp* beginnen.

Fußnoten

(1)

Nebenbei bemerkt ist das üblicherweise von TCP/IP verwendete Ethernet-Protokoll nicht ganz mit IEEE 802.3 identisch. Ethernet-Frames verwenden ein Typ-Feld, wo IEEE 802.3-Frames mit einem Längen-Feld arbeiten.

Inhaltsverzeichnis  **Anhang D**   **Bibliographie**

Bibliographie

Verwandte Literatur

[Computer Networks] Andrew S. Tanenbaum: *Computer Networks*. Prentice Hall International, 1989.

Dieses Buch bietet einen sehr guten Einblick in allgemeine Netzwerk-Aspekte. Ausgehend vom OSI-Referenzmodell werden Design-Aspekte jeder Schicht sowie die zur Lösung verwendeten Algorithmen erklärt. Für jede Schicht werden die Implementierungen verschiedener Netzwerke, darunter auch des ARPANET, untereinander verglichen. Leider macht es die Unmenge der verwendeten Abkürzungen manchmal schwierig, dem Text zu folgen.

[Connecting] Susan Estrada: *Connecting to the Internet: An O'Reilly Buyer's Guide*. O'Reilly & Associates, 1993.

Enthält praktische Ratschläge, wie Sie sich Ihren Internet-Provider aussuchen, und welche Arten von Internet-Verbindungen angeboten werden. Es hilft Ihnen bei der Entscheidung, welche Dienste Sie benötigen und vergleicht die unterschiedlichen Kostenstrukturen verschiedener Anbieter. Das Buch enthält eine internationale Liste mit Internet-Providern.

[DNS] Paul Albitz, Cricket Liu: *DNS and BIND*. O'Reilly & Associates, 1996.

Deutsche übersetzung *DNS und Bind* O'Reilly Verlag, 1997.

Ein nützliches Buch für alle, die den DNS-Name-Service verwalten müssen. Alle DNS-Features werden sehr ausführlich behandelt. Das Buch enthält Beispiele, die sogar die BIND-Optionen verständlich machen, die auf den ersten Blick ziemlich sonderlich erscheinen. Ich fand es sehr gut zu lesen und habe viel gelernt.

[Expect] Don Libes: *Exploring Expect: A Tcl-based Toolkit for Automating Interactive Programs*. O'Reilly & Associates, 1995.

[Installation] Matt Welsh: *Installation and Getting Started Guide*. Linux Documentation Project

Dieses Buch hilft Ihnen dabei, eine Linux-Distribution auf Ihrem System zum Laufen zu bringen. Es behandelt alle während der Installation auftretenden Arbeiten, von den Grundlagen der Erzeugung eines Dateisystems bis hin zur einfachen Systemadministration.

[Internetworking] Douglas R. Comer: *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture*. Prentice Hall International, 1991.

[Linux-Handbuch] Sebastian Hetze, Dirk Hohndel, Martin Müller, Olaf Kirch: *LinuX Anwenderhandbuch und Leitfaden für die Systemverwaltung*. LunetIX-Verlag, 1994.

Das Buch enthält alle notwendigen Informationen für die Installation einer Linux-Distribution. Insbesondere die Partitionierung der Festplatte und die Einrichtung eines Linux-Dateisystems werden ausführlich erklärt.

[Linux-Workstation] Stefan Strobel, Thomas Uhl: *LINUX. Vom PC zur Workstation.* Springer-Verlag, 1995.

Das Buch führt in die für Linux verfügbaren Pakete ein und enthält eine Anleitung zur Installation und Konfiguration. Schwerpunkte bilden dabei die graphische Oberfläche, die Netzwerkfähigkeit und die Tools des GNU-Projekts.

[Managing UUCP] Tim O'Reilly, Grace Todino: *Using and Managing UUCP* O'Reilly & Associates, 1996.

Das Standardbuch zu UUCP-Netzwerken. Behandelt wird UUCP Version 2 ebenso wie BNU. Es hilft von Beginn an bei der Einrichtung eines UUCP-Node. Praktische Tips und Lösungen für viele Probleme, wie beispielsweise die Überprüfung von Verbindungen oder das Schreiben guter Chat-Scripten, sind hier zu finden. Auch exotischere Themen wie »wandernde« UUCP-Nodes oder die kleinen Unterschiede verschiedener UUCP-Implementierungen werden behandelt.

[NFS und NIS] Hal Stern: *Verwaltung von UNIX-Netzwerken mit NFS und NIS.* O'Reilly Verlag, 1995. US-Originalausgabe: *Managing NFS and NIS*, 1992.

Eine ideale Ergänzung zum TCP/IP-Buch von Craig Hunt. Behandelt ausführlich die Verwendung des NIS (»Network Information System«) und des NFS (»Network File System«), einschließlich der Konfiguration eines Automounters und PC/NFS.

[NIS+] Rick Ramsey: *All about Administering NIS+*. Prentice Hall, 1993.

[Running Linux] Matt Welsh: *Linux -- Wegweiser zur Installation & Konfiguration*. 2. Auflage, O'Reilly Verlag, 1997. US-Originalausgabe: *Running Linux*. 1995.

Ein Buch, das auf dem *Installation and Getting Started Guide* desselben Autors aufbaut, aber noch etwas ausführlicher ist. Es enthält weiterführende Informationen zu kostenlos verfügbaren Software-Tools, die mit Linux-Distributionen angeboten werden.

[sendmail] Costales, Bryan, with Eric Allman and Neil Rickert: *sendmail*. O'Reilly & Associates, 1996.

Ein ausführliches Buch, das die Installation dieses Paketes behandelt. Kann sowohl als Einführung und als Referenz verwendet werden.

[TCP/IP] Craig Hunt: *TCP/IP Netzwerk Administration*. O'Reilly Verlag, 1994. US-Originalausgabe: *TCP/IP Network Administration*, 1992.

Wenn Ihnen der *Linux* -- *Wegweiser für Netzwerker* nicht genügt, sollten Sie sich dieses Buch beschaffen. Es behandelt alle Themen von der Zuweisung einer IP-Adresse über die Fehlersuche bis hin zu Sicherheitsaspekten. Themenschwerpunkt ist die Einrichtung von TCP/IP, d. h. Interface-Konfiguration, Einrichten des Routens und die Auflösung von Namen. Enthält eine ausführliche Beschreibung der Fähigkeiten der Routing-Dämonen *routed* und *gated*, die dynamisches Routen ermöglichen.

Darüber hinaus werden die Konfiguration von Anwendungsprogrammen und Netzwerk-Dämon

wie *inetd* sowie die *r*-Befehle, NIS und NFS beschrieben. Der Anhang enthält ausführliche Referenzen zu *gated* und *named* sowie eine Beschreibung der Konfiguration des Berkeley *sendmail*-Programms.

[UNIX Network Programming] Richard W. Stevens: *UNIX Network Programming*. Prentice Hall International, 1990.

Das wahrscheinlich am häufigsten verwendete Buch über die Programmierung von TCP/IP-Netzwerken. Behandelt gleichzeitig viele Fallstricke des Internet-Protokolls.(1)

[UNIX Security] Simson Garfinkel, Gene Spafford: *Practical UNIX & Internet Security*. O'Reilly & Associates, 1995.

Ein Muß für jeden, der Systeme mit Netzwerkzugriff verwaltet, aber auch für andere Leser sinnvoll. Das Buch behandelt alle Themen der Computersicherheit, angefangen bei den grundlegenden UNIX-Sicherheitsfeatures, bis hin zur physikalischen Sicherheit. Obwohl Sie bestrebt sein sollten, alle Teile Ihres Systems entsprechend zu schützen, ist die Diskussion von Sicherheit in Netzwerken aus Sicht dieses Buches wohl am interessantesten. Neben grundlegenden Sicherheitsregeln für Berkeley-Dienste (*telnet*, *rlogin* etc.), NFS und NIS, werden auch Sicherheitsfeatures wie Kerberos vom MIT, »Secure RPC« von Sun und die Verwendung von Firewalls zur Abschirmung Ihres Netzwerks vor Angriffen aus dem Internet erläutert.

[Whole Internet] Ed Krol: *Die Welt des Internet -- Handbuch und Übersicht*. O'Reilly Thomson Verlag, 1995. US-Originalausgabe *: The Whole Internet User's Guide & Catalog, 2nd Ed.*, 1994.

Ein »Handbuch« zu allen Diensten des Internet. Enthält ausführliche Anweisungen zu deren Verwendung, sowie eine Liste populärer Server.

[Zen] Brendan P. Kehoe: *Zen und die Kunst des Internet*

»Zen« war wahrscheinlich *das* erste Internet-Handbuch. Es führt den neuen Benutzer in die verschiedenen Dienste, Gebräuche und Traditionen des Internet ein. Auf über 100 Seiten werden Themen behandelt wie E-Mail, Usenet-News bis hin zum Internet-Wurm. Das Buch liegt auf vielen FTP-Servern vor, kann über »anonymous FTP« heruntergeladen, gedruckt und frei verteilt werden. Ist in gedruckter (und erweiterter) Form bei Prentice Hall erschienen (1994, ISBN: 3-930436-06-X).

HOWTOs

Nachfolgend ein ins Deutsche übersetzter Auszug aus dem HOWTO-INDEX, Version 2.0 (17 März 1994), geschrieben von Matt Welsh.

Was sind Linux-HOWTOs? Linux-HOWTOs sind kurze Online-Dokumente, die verschiedene Aspekte der Konfiguration und der Verwendung des Linux-Systems beschreiben. So existiert beispielsweise ein Installations-HOWTO, das Anweisungen zur Installation von Linux enthält, und ein Mail-HOWTO, das beschreibt, wie Sie Mail unter Linux einrichten und konfigurieren. Weitere Beispiele sind das NET-2-HOWTO (früher NET-2-FAQ) und das Printing-HOWTO. Informationen in HOWTOs sind üblicherweise detaillierter und tiefschürfender als die eigentliche Linux-FAQ. Aus diesem Grund wird die Linux-FAQ umgeschrieben. Ein großer Teil der Informationen wird danach in den verschiedenen HOWTO-Dokumenten zu finden sein. Die FAQ wird dann eine kürzere Liste häufig gestellter Fragen (»Frequently Asked Questions«) über Linux enthalten, in der kleinere, spezifische Themen besprochen

werden. Die meisten »sinnvollen« Informationen der FAQ werden dann in den HOWTOs behandelt.

HOWTOs sind, ähnlich wie FAQs, umfassende Dokumente, liegen aber nicht in Form von Fragen und Antworten vor. Trotzdem enthalten viele HOWTOs einen FAQ-Abschnitt am Ende des Dokuments. Beispielsweise wurde die NET-2-FAQ in NET-2-HOWTO umbenannt, weil sie nicht mehr die Frage-und-Antwort-Form hatte. An vielen Orten ist das NET-2-HOWTO aber immer noch unter dem Namen NET-2-FAQ zu finden. Es handelt sich trotzdem um ein und dasselbe Dokument.

Sie können HOWTOs über »anonymous FTP« von den folgenden Servern sowie von vielen Rechnern, die diese Daten spiegeln, herunterladen. Diese Server sind in der Linux-META-FAQ (s. u.) aufgeführt.

```
sunsite.unc.edu:/pub/Linux/docs/HOWTO
tsx-11.mit.edu:/pub/linux/docs/HOWTO
```

Der nachfolgend aufgeführte Index enthält eine Liste aller verfügbaren HOWTOs. HOWTOs werden regelmäßig in den Newsgruppen *comp.os.linux* und *comp.os.linux.announce* gepostet. Zusätzlich wird eine Reihe von HOWTOs in *news.answers* crossgepostet. Aus diesem Grund finden Sie die Linux-HOWTOs auf dem *news.answers*-Archiv-Server **rtfm.mit.edu**.

Momentan sind die folgenden Linux-HOWTOs verfügbar.

[Busmouse HOWTO] Mike Battersby: *Linux Busmouse HOWTO*

Informationen zur Busmaus-Kompatibilität unter Linux.

[CDROM HOWTO] Jeff Tranter: *Linux CDROM HOWTO*

Informationen zur Kompatibilität von CD-ROM-Laufwerken mit Linux.

[Distribution HOWTO] Matt Welsh: *Linux Distribution HOWTO*

Eine Liste mit Mail-Order-Distributionen und anderen kommerziellen Diensten.

[DOSEMU HOWTO] Michael E. Deisher: *Linux DOSEMU HOWTO*

HOWTO zum Linux-MS-DOS Emulator (DOSEMU).

[Ethernet HOWTO] Paul Gortmaker: *Linux Ethernet HOWTO*

Informationen zur Kompatibilität von Ethernet-Hardware unter Linux.

[Ftape HOWTO] Linux ftape-HOWTO maintainer: *Linux Ftape HOWTO*

Informationen zur Kompatibilität von Bandlaufwerken unter Linux.

[Hardware HOWTO] Ed Carp: *Linux Hardware Compatibility HOWTO*

Eine nahezu vollständige Liste von Hardware-Komponenten, die mit Linux laufen.

[Index HOWTO] Matt Welsh: *Linux HOWTO Index*

Index von HOWTO-Dokumenten zu Linux.

[Installation HOWTO] Matt Welsh: *Linux Installation HOWTO*

Wie beschaffe und installiere ich Linux-Software?

[JE HOWTO] Yasuhiro Yamazaki: *Linux JE-HOWTO*

Informationen zu JE, einer Reihe von Linux-Erweiterungen für die japanische Sprache.

[Keystroke HOWTO] Zenon Fortuna: *Linux Keystroke HOWTO*

Wie binde ich unter Linux Makros an bestimmte Tasten?

[MGR HOWTO] Vincent Broman: *Linux MGR HOWTO*

Informationen zum MGR-Grafikinterface unter Linux.

[email HOWTO] Vince Skahan: *Linux Electronic Mail HOWTO*

Informationen zu Linux-basierten Mail-Servern und Clients.

[NET-2 HOWTO] Terry Dawson: *Linux NET-2 HOWTO*

Wie werden TCP/IP-Netzwerke, SLIP, PLIP und PPP unter Linux konfiguriert?

[News HOWTO] Vince Skahan: *Linux News HOWTO*

Informationen zu Usenet-News-Server- und Client-Software für Linux.

[NIS HOWTO] Andrea Dell'Amico, Mitchum DSouza, Erwin Embsen: *The Linux NIS(YP)/NIS+/NYS HOWTO*

Einführung und Installations-Informationen zu NIS-Paketen.

[PCI HOWTO] Michael Will: *Linux PCI-HOWTO*

Informationen zur Kompatibilität der PCI-Architektur mit Linux.

[Printing HOWTO] Grant Taylor: *Linux Printing HOWTO*

HOWTO zu Druck-Software für Linux.

[SCSI HOWTO] Drew Eckhardt: *Linux SCSI HOWTO*

Informationen zur Kompatibilität von SCSI-Geräten mit Linux.

[Serial HOWTO] Greg Hankins: *Linux Serial HOWTO*

Informationen zur Verwendung serieller Einheiten und Kommunikations-Software.

[Sound HOWTO] Jeff Tranter: *Linux Sound HOWTO*

Sound-Hardware und -Software für das Linux-Betriebssystem.

[Term HOWTO] Bill Reynolds: *Linux Term HOWTO*

HOWTO zur Verwendung des »term«-Kommunikationspaketes auf Linux-Systemen.

[Tips HOWTO] Vince Reed: *Linux Tips HOWTO*

HOWTO mit verschiedenen Tips und Tricks für Linux.

[UUCP HOWTO] Vince Skahan: *Linux UUCP HOWTO*

Informationen zu UUCP-Software für Linux.

[XFree86 HOWTO] Helmut Geyer: *XFree86 HOWTO*

HOWTO zur Installation von XFree86.

RFCs

RFC 1597. Rekhter Y. Watson T.J., et al: *Address Allocation for Private Internets*

Dieses RFC enthält eine Liste der IP-Netzwerk-Adressen, die von privaten Organisationen intern verwendet werden können, ohne daß diese Nummern der »Internet Assigned Numbers Authority« (IANA) bekanntgegeben werden müssen. Das Dokument behandelt auch die Vor- und Nachteile der Verwendung dieser Nummern.

RFC 1340. Postel J., Reynolds J.: *Assigned Numbers*

Das Assigned Numbers RFC definiert die Bedeutung der von verschiedenen Protokollen verwendeten Nummern., z. B. die Port-Nummern, an denen Standard-TCP- und UDP-Server »horchen« sowie die im IP-Datagramm-Header verwendeten Protokoll-Nummern.

RFC 1144. Jacobson V.: *Compressing TCP/IP headers for low-speed serial links*

Dieses Dokument beschreibt den zur Kompression von TCP/IP-Headern bei CLISP und PPP verwendeten Algorithmus. Lohnende Lektüre!

RFC 1033. Lottor M.: *Domain Administrators Operations Guide*

Zusammen mit RFC 1034 und RFC 1035 die definitive Quelle für DNS, das Domain Name System.

RFC 1034. Mockapetris P. V.: *Domain Names* -- *Concepts and Facilities*

Begleitdokument zu RFC 1033.

RFC 1035. Mockapetris P. V.: *Domain names* -- *Implementation and Specification*

Begleitdokument zu RFC 1033.

RFC 974. Partridge C.: *Mail Routing and the Domain System*

Dieses Dokument beschreibt das Mail-Routing im Internet. Hier finden Sie die ganze Wahrheit über MX-Records.

RFC 1548. Simpson W. A.: *The Point-to-Point Protocol (PPP)*

Der Standard für die PPP-Methode, Datagramme über Point-to-Point-Links zu transportieren.

RFC 977. Kantor B. Lapsley P.: *Network News Transfer Protocol*

Die Definition von NNTP, dem gängigen News-Transport im Internet.

RFC 1094. Nowicki B.: *NFS: Network File System Protocol specification*

Die formale Spezifikation der NFS- und Mount-Protokolle (Version 2).

RFC 1055. Romkey J. L.: *Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP*

Beschreibt SLIP, das »Serial Line Internet Protocol«.

RFC 1057. Sun Microsystems, Inc.: *RPC: Remote Procedure Call Protocol Specification: Version 2*

RFC 1058. Hedrick C. L.: *Routing Information Protocol*

Beschreibt RIP, das verwendet wird, um dynamische Routing-Informationen innerhalb von LANs und MANs auszutauschen.

RFC 1535. Gavron E.: *A Security Problem and Proposed Correction with Widely Deployed DNS Software*

Dieses RFC diskutiert ein Sicherheitsproblem mit der voreingestellten Suchliste, die von älteren Versionen der BIND-Resolver-Library verwendet wird.

RFC 821. Postel J. B.: *Simple Mail Transfer Protocol*

Definiert SMTP, das Mail-Transport-Protokoll über TCP/IP.

RFC 1036. Adams R., Horton M. R.: *Standard for the Interchange of USENET messages*

Dieses RFC beschreibt das von Usenet-News-Messages verwendete Format, und wie diese über das Internet und über UUCP-Netzwerke ausgetauscht werden. Eine überarbeitete Fassung dieses RFC wird in naher Zukunft erwartet.

RFC 822. Crocker D.: *Standard for the Format of ARPA Internet text messages*

Die definitive Quelle der Weisheit für RFC-konforme Mail. Jeder kennt es, aber nur wenige haben es gelesen.

RFC 968. Cerf V.: *Twas the Night Before Start-up*

Wer sagt, daß die Helden der Netzwerke nicht besungen werden?

Fußnoten

(1)

Stevens hat gerade bei Addison Wesley ein neues Buch namens TCP/IP Illustrated, Volume 1, The Protocols herausgebracht, das die Protokolle der TCP/IP-Familie bis in die kleinsten Details behandelt.

Inhaltsverzeichnis  **Glossar** 